IN THE UNITED STATES DISTRICT COURT

FOR THE DISTRICT OF DELAWARE

FINJAN SOFTWARE, LTD., an Israel corporation,	)	
	)	
Plaintiff,	)	Civil Action No. 06-369-GMS
	)	
v.	)	
	)	
SECURE COMPUTING CORPORATION, a Delaware corporation; CYBERGUARD CORPORATION, a Delaware corporation, WEBWASHER AG, a German corporation and DOES 1 THROUGH 100,	)	**DEMAND FOR JURY TRIAL**
	)	
Defendants.	)	

AMENDED COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Finjan Software, Ltd. ("Plaintiff" or "Finjan") alleges as follows:

THE PARTIES

1. Plaintiff Finjan is a corporation organized and existing under the laws of Israel, with its principal place of business at Hamachshev St. 1, New Industrial Area, Netanya, 42504, Israel.

2. On information and belief, Defendant Secure Computing Corporation ("Secure Computing") is a corporation organized and existing under the laws of the State of Delaware, with its corporate headquarters at 4810 Harwood Road, San Jose, California 95124.

3. On information and belief, Defendant Cyberguard Corporation ("Cyberguard") is a corporation organized and existing under the laws of the State of Delaware and a wholly owned subsidiary of Secure Computing, with its corporate headquarters at 4810 Harwood Road, San Jose, California 95124.

4. On information and belief, Defendant Webwasher AG ("Webwasher") is a corporation organized and existing under the laws of Germany and a wholly owned subsidiary of Cyberguard, with a branch office at 5201 Great America Parkway, Suite 432, Santa Clara, CA 95054.

JURISDICTION AND VENUE

5. This action arises under the Patent Act, 35 U.S.C. §271 *et seq.* This Court has original jurisdiction over this controversy pursuant to 28 U.S.C. §§1331 and 1338.

6. Venue in this judicial district is proper under 28 U.S.C. §§ 1391 (b) and (c) and/or 28 U.S.C. § 1400(b). Personal jurisdiction over defendants comports with the United States Constitution and § 3104 of the Delaware Code because defendants are Delaware corporations and/or have and continue to infringe, contributorily infringe and/or induce the infringement of U.S. Patent No. 6,092,194, U.S. Patent No. 6,804,780 and U.S. Patent No. 7,058,822 in this district.

PLAINTIFF'S PATENT

7. On July 18, 2000, United States Patent No. 6,092,194 ("the '194 Patent"), entitled SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES, was issued to Shlomo Touboul. Finjan was assigned all ownership rights to the '194 Patent. A true and correct copy of the '194 Patent is attached to this complaint as Exhibit A and is incorporated by reference herein.

8. On October 12, 2004, United States Patent No. 6,804,780 ("the '780 Patent"), entitled SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES, was issued to Shlomo Touboul. Finjan was assigned all

ownership rights to the '780 Patent. A true and correct copy of the '780 Patent is attached to this complaint as Exhibit B and is incorporated by reference herein.

9. On June 6, 2006, United States Patent No. 7,058,822 ("the '822 Patent"), entitled MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS, was issued to Yigal Mordechai Edery, Nimrod Itzhak Vered and David R. Kroll. Finjan was assigned all ownership rights to the '822 Patent. A true and correct copy of the '822 Patent is attached to this complaint as Exhibit C and is incorporated by reference herein.

10. The '194 Patent, ‘780 Patent and '822 Patent are directed to a system and method for protecting networks and computers from hostile downloadable executable application programs.

PATENT INFRINGEMENT

11. On information and belief, defendant Secure Computing is in the business of developing and distributing network and systems security solutions to organizations. Secure Computing has and continues to infringe the '194 Patent, ‘780 Patent and '822 Patent by making, using, selling, distributing, advertising and marketing products, including but not limited to the Webwasher Secure Content Management ("SCM") suite, that infringe the '194 Patent, ‘780 Patent and '822 Patent.

12. On information and belief, defendant Cyberguard is in the business of developing and distributing information security solutions. Cyberguard has and continues to infringe the '194 Patent, ‘780 Patent and '822 Patent by making, using, selling, distributing, advertising and marketing products, including but not limited to the Webwasher Secure Content Management ("SCM") suite, that infringe the '194 Patent, ‘780 Patent and '822 Patent.

13. On information and belief, defendant Webwasher is in the business of developing and distributing Internet and email content security and filtering solutions. Webwasher has and

continues to infringe the '194 Patent, ‘780 Patent and '822 Patent by making, using, selling, distributing, advertising and marketing products, including but not limited to the Webwasher Secure Content Management ("SCM") suite, that infringe the '194 Patent, ‘780 Patent and '822 Patent.

FIRST CAUSE OF ACTION

(Infringement of the '194 Patent)

14. Finjan realleges each and every allegation set forth in Paragraphs 1 through 13, inclusive, and incorporates them herein by reference.

15. Defendants Secure Computing, Cyberguard and Webwasher have been and continue to infringe, contributorily infringe, and/or induce the infringement of the '194 Patent by making, using, selling and/or offering to sell products which infringe the '194 Patent, including but not limited to the Webwasher SCM suite, and will continue to do so until enjoined by this Court.

16. Defendants' infringement of the '194 Patent has been and continues to be willful and deliberate.

17. Defendants' infringement of the '194 Patent has injured and continues to injure Finjan in an amount to be proven at trial.

SECOND CAUSE OF ACTION

(Infringement of the '780 Patent)

18. Finjan realleges each and every allegation set forth in Paragraphs 1 through 17, inclusive, and incorporates them herein by reference.

19. Defendants Secure Computing, Cyberguard and Webwasher have been and continue to infringe, contributorily infringe, and/or induce the infringement of the '780 Patent by making, using, selling and/or offering to sell products which infringe the '780 Patent, including but not limited to the Webwasher SCM suite, and will continue to do so until enjoined by this Court.

20. Defendants' infringement of the '780 Patent has been and continues to be willful and deliberate.

21. Defendants' infringement of the '780 Patent has injured and continues to injure Finjan in an amount to be proven at trial.

THIRD CAUSE OF ACTION

(Infringement of the '822 Patent)

22. Finjan realleges each and every allegation set forth in Paragraphs 1 through 21, inclusive, and incorporates them herein by reference.

23. Defendants Secure Computing, Cyberguard and Webwasher have been and continue to infringe, contributorily infringe, and/or induce the infringement of the '822 Patent by making, using, selling and/or offering to sell products which infringe the '822 Patent, including but not limited to the Webwasher SCM suite, and will continue to do so until enjoined by this Court.

24. Defendants' infringement of the '822 Patent has been and continues to be willful and deliberate.

25. Defendants' infringement of the '822 Patent has injured and continues to injure Finjan in an amount to be proven at trial.

PRAYER FOR RELIEF

WHEREFORE, Finjan prays that the Court grant the following relief and judgment:

A. A preliminary and permanent injunction against Defendants Secure Computing, Cyberguard and Webwasher and its respective officers, employees, agents, servants, attorneys, instrumentalities, and/or those in privity with them, from infringing, contributorily infringing, or inducing the infringement of the '194 Patent, ‘780 Patent and '822 Patent, and for all further and proper injunctive relief pursuant to 35 U.S.C. §283;

B. An award to Finjan of such damages as it shall prove at trial against Defendants Secure Computing, Cyberguard and Webwasher, that are adequate to fully compensate it for their infringement of the '194 Patent, ‘780 Patent and '822 Patent, said damages to be no less than a reasonable royalty;

C. An award to Finjan for willful infringement against Defendants Secure Computing, Cyberguard and Webwasher of three times the damages so determined, as provided by 35 U.S.C. §284, together with prejudgment interest from the date infringement of the '194 Patent, ‘780 Patent and/or '822 Patent began;

D. A finding that this case is "exceptional" and an award to Finjan of its costs and reasonable attorney's fees, as provided by 35 U.S.C. §285;

E. Such further and other relief as the Court and/or jury may deem proper and just.

DEMAND FOR JURY TRIAL

Plaintiff Finjan Software, Ltd. hereby demands a trial by jury on all issues triable by a jury.

POTTER ANDERSON & CORROON LLP

OF COUNSEL:

Paul J. Andre
Perkins Coie LLP
101 Jefferson Drive
Menlo Park, California 94025-1114
(650) 838-4300

Dated: April 5, 2007
787724

By /s/ Philip A. Rovner
Philip A. Rovner (#3215)
Hercules Plaza
P. O. Box 951
Wilmington, DE 19899
(302) 984-6000
provner@potteranderson.com

Attorneys for Plaintiff
Finjan Software, Ltd.

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

CERTIFICATE OF SERVICE

I, Philip A. Rovner, hereby certify that on April 10, 2007, the within document was filed with the Clerk of the Court using CM/ECF which will send notification of such filing(s) to the following; that the document was served on the following counsel as indicated; and that the document is available for viewing and downloading from CM/ECF.

BY HAND DELIVERY

Frederick L. Cottrell, III, Esq.
Gregory E. Stuhlman, Esq.
Richards, Layton & Finger, P.A.
One Rodney Square
920 N. King Street
Wilmington, DE 19801
cottrell@rlf.com;
stuhlman@rlf.com

I hereby certify that on April 10, 2007 I have sent by Federal Express the foregoing document to the following non-registered participants:

Jake M. Holdreith, Esq.
Christopher A. Seidl, Esq.
Robins, Kaplan, Miller & Ciresi L.L.P.
2800 LaSalle Plaza
800 LaSalle Avenue
Minneapolis, MN 55402
jmholdreith@rkmc.com
caseidl@rkmc.com

/s/ Philip A. Rovner
Philip A. Rovner (#3215)
Potter Anderson & Corroon LLP
Hercules Plaza
P.O. Box 951
Wilmington, Delaware 19899
(302) 984-6000
E-mail: provner@potteranderson.com

EXHIBIT A

US006092194A

United States Patent [19]

Touboul

[11] **Patent Number: 6,092,194**

[45] **Date of Patent: *Jul. 18, 2000**

[54] **SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES**

[75] Inventor: **Shlomo Touboul**, Kefar-Haim, Israel

[73] Assignee: **Finjan Software, Ltd.**, Netanya, Israel

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **08/964,388**

[22] Filed: **Nov. 6, 1997**

Related U.S. Application Data

[60] Provisional application No. 60/030,639, Nov. 8, 1996.

[51] **Int. Cl.**^{7} **H04L 1/00**

[52] **U.S. Cl.** **713/200**

[58] **Field of Search** 395/187.01, 186; 713/200, 201, 202; 714/38, 704; 709/229

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,077,677	12/1991	Murphy et al.	395/10
5,361,359	11/1994	Tajalli et al.	395/700
5,485,409	1/1996	Gupta et al.	395/186
5,485,575	1/1996	Chess et al.	395/183.14
5,572,643	11/1996	Judson	395/793
5,623,600	4/1997	Ji et al.	395/187.01
5,638,446	6/1997	Rubin	380/25
5,692,047	11/1997	McManis	380/4
5,692,124	11/1997	Holden et al.	395/187.01
5,720,033	2/1998	Deo	395/186
5,724,425	3/1998	Chang et al.	380/25
5,740,248	4/1998	Fieres et al.	380/25
5,761,421	6/1998	van Hoff et al.	395/200.53
5,765,205	6/1998	Breslau et al.	711/203
5,784,459	7/1998	Devarakonda et al.	380/4
5,796,952	8/1998	Davis et al.	395/200.54
5,805,829	9/1998	Cohen et al.	395/200.32
5,832,208	11/1998	Chen et al.	395/187.01
5,850,559	12/1998	Angelo et al.	395/750.03
5,864,683	1/1999	Boebert et al.	395/200.79
5,892,904	4/1999	Atkinson et al.	395/187.01

OTHER PUBLICATIONS

Web page: http://iel.ihs.com:80/cgi–bin/iel_cgi?se...2ehts%26ViewTemplate%3ddocvie%5fb%2ehts, Okamato, E. et al., "ID–Based Authentication System For Computer Virus Detection", IEEE/IEE Electronic Library online, Electronics Letters, vol. 26, Issue 15, ISSN 0013–5194, Jul. 19, 1990, Abstract and pp. 1169–1170.

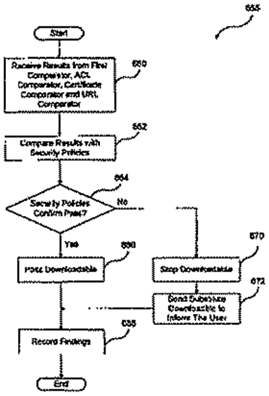
(List continued on next page.)

Primary Examiner—Robert W. Beausoliel, Jr.
Assistant Examiner—Christopher Revak
Attorney, Agent, or Firm—Graham & James LLP

[57] **ABSTRACT**

A system protects a computer from suspicious Downloadables. The system comprises a security policy, an interface for receiving a Downloadable, and a comparator, coupled to the interface, for applying the security policy to the Downloadable to determine if the security policy has been violated. The Downloadable may include a Java™ applet, an ActiveX™ control, a JavaScript™ script, or a Visual Basic script. The security policy may include a default security policy to be applied regardless of the client to whom the Downloadable is addressed, or a specific security policy to be applied based on the client or the group to which the client belongs. The system uses an ID generator to compute a Downloadable ID identifying the Downloadable, preferably, by fetching all components of the Downloadable and performing a hashing function on the Downloadable including the fetched components. Further, the security policy may indicate several tests to perform, including (1) a comparison with known hostile and non-hostile Downloadables; (2) a comparison with Downloadables to be blocked or allowed per administrative override; (3) a comparison of the Downloadable security profile data against access control lists; (4) a comparison of a certificate embodied in the Downloadable against trusted certificates; and (5) a comparison of the URL from which the Downloadable originated against trusted and untrusted URLs. Based on these tests, a logical engine can determine whether to allow or block the Downloadable.

68 Claims, 10 Drawing Sheets


655
Start
Receive Results from First Comparator, ACL Comparator, Certificate Comparator and URL Comparator
660
Compare Results with Security Policies
662
664
Security Policies Confirm Pass?
No
Yes
666
Pass Downloadable
670
Stop Downloadable
672
Send Substitute Downloadable to Inform The User
668
Record Findings
End

OTHER PUBLICATIONS

"Finjan Announces a Personal Java ™ Firewall For Web Browsers—the SurfinShield™ 1.6", Press Release of Finjan Releases SurfinShield, Oct. 21, 1996, 2 pages.

"Finjan Software Releases SurfinBoard, Industry's First JAVA Security Product For the World Wide Web", Article published on the Internet by Finjan Software, Ltd., Jul. 29, 1996, 1 page.

"Powerful PC Security for the New World of Java™ and Downloadables, Surfin Shield™"Article published on the Internet by Finjan Software Ltd., 1996, 2 Pages.

"Company Profile Finjan—Safe Surfing, The Java Security Solutions Provider" Article published on the Internet by Finjan Software Ltd., Oct. 31, 1996, 3 pages.

"Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0" Las Vegas Convention Center/Pavillion 5 P5551, Nov. 18, 1996, 3 pages.

"Java Security: Issues & Solutions" Article published on the Internet by Finjan Software Ltd., 1996, 8 pages.

"Products" Article published on the Internet, 7 pages.

Mark LaDue, "Online Business Consultant" Article published on the Internet, Home Page, Inc. 1996, 4 pages.

Jim K. Omura, "Novel Applications of Cryptography in Digital Communications", IEEE Communications Magazine, p 27, May 1990.

Norvin Leach et al, "IE 3.0 applets will earn certification", PC Week, v13, n29, p1(2), Jul. 1996.

Microsoft Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet", Microsoft Corporation, Oct. 1996.

Frequently Asked Questions About Authenticode, Microsoft Corporation, Feb. 1997.

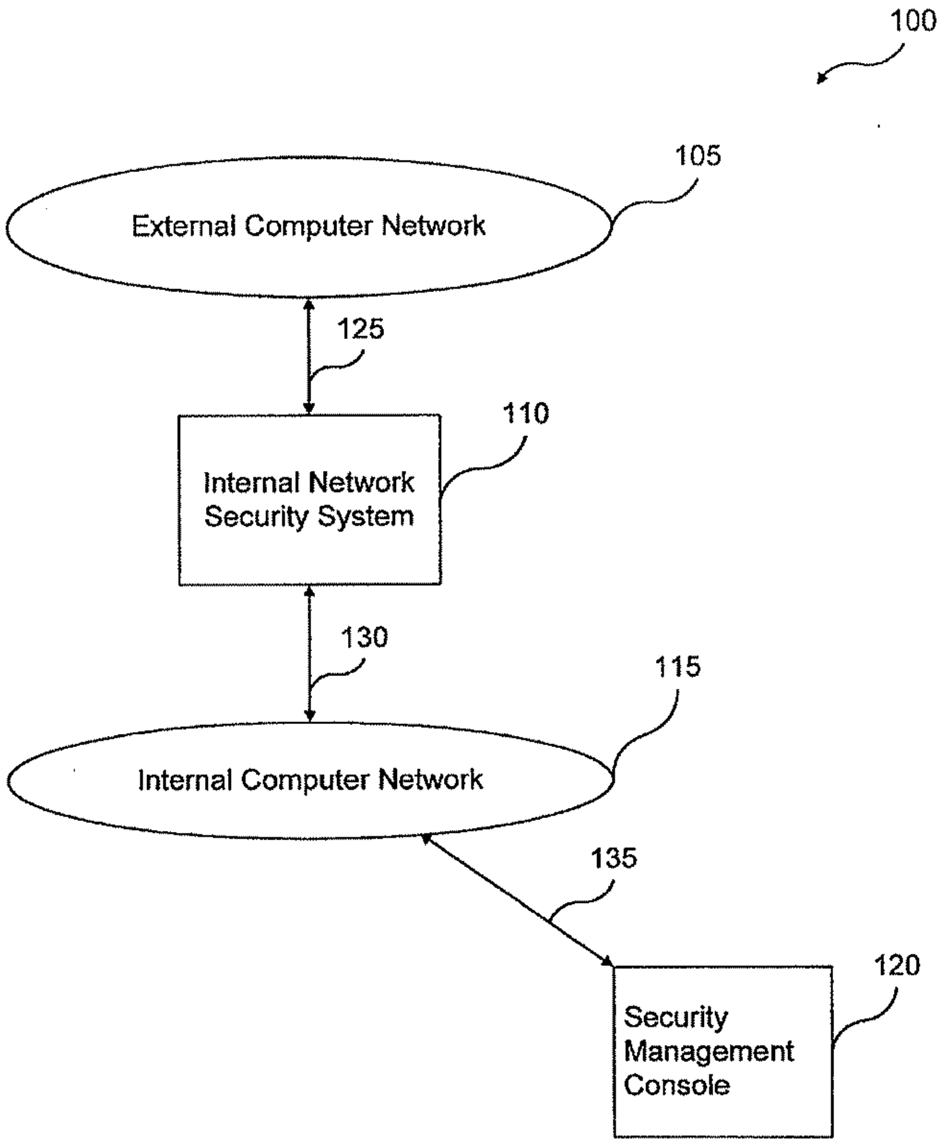

100
105
External Computer Network
125
110
Internal Network Security System
130
115
Internal Computer Network
135
120
Security Management Console

FIG. 1

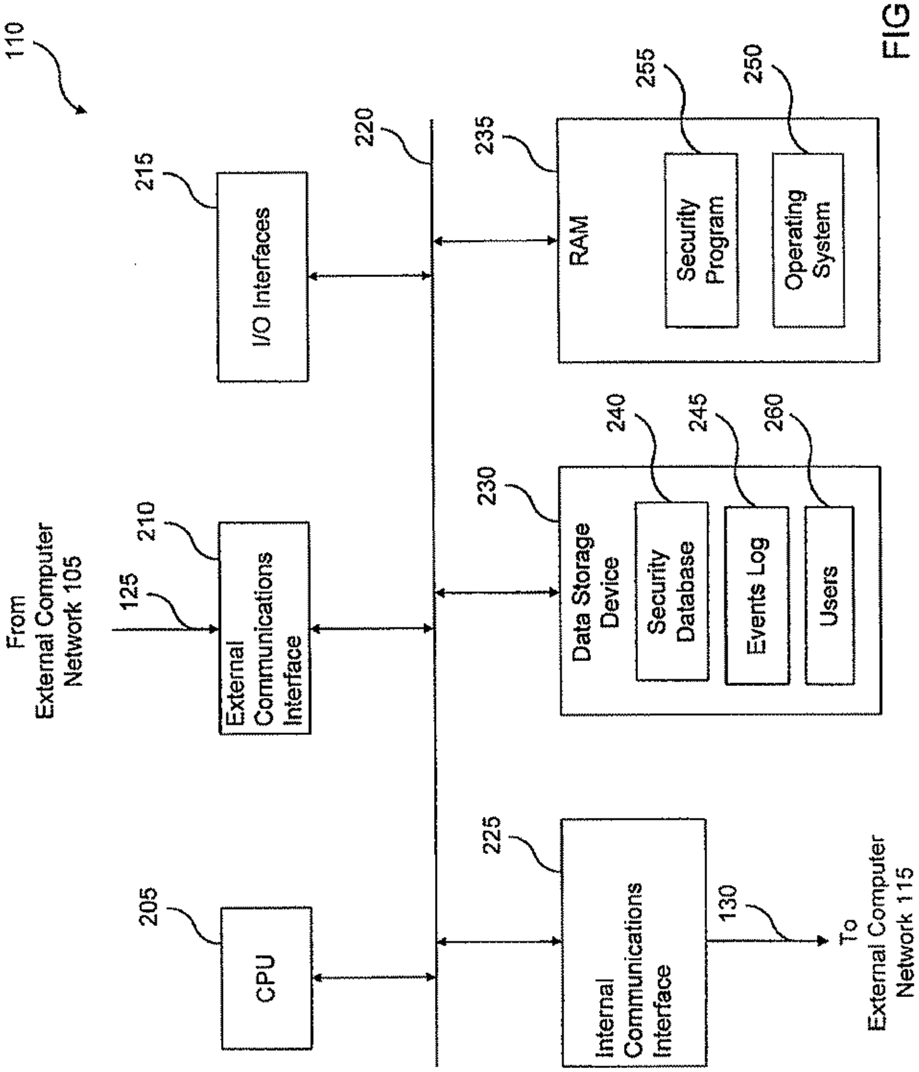

110
215
I/O Interfaces
220
235
RAM
255
Security Program
250
Operating System
From External Computer Network 105
125
210
External Communications Interface
230
Data Storage Device
240
Security Database
245
Events Log
260
Users
205
CPU
225
Internal Communications Interface
130
To External Computer Network 115

FIG. 2

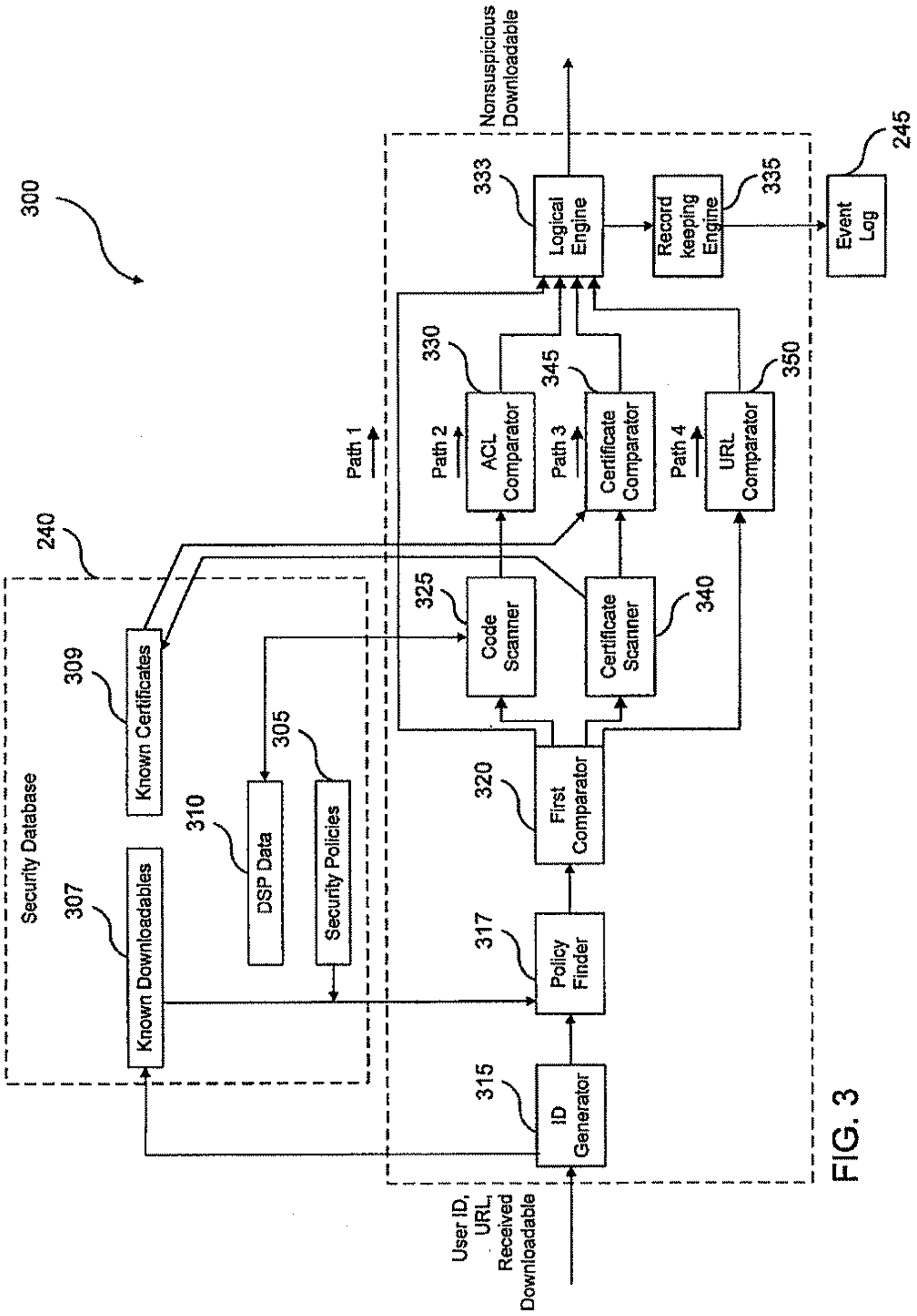

300
240
Security Database
307
Known Downloadables
309
Known Certificates
310
DSP Data
305
Security Policies
User ID,
URL,
Received
Downloadable
315
ID
Generator
317
Policy
Finder
320
First
Comparator
Path 1
Path 2
Path 3
Path 4
325
Code
Scanner
340
Certificate
Scanner
330
ACL
Comparator
345
Certificate
Comparator
350
URL
Comparator
333
Logical
Engine
335
Record
keeping
Engine
245
Event
Log
Nonsuspicious
Downloadable

FIG. 3

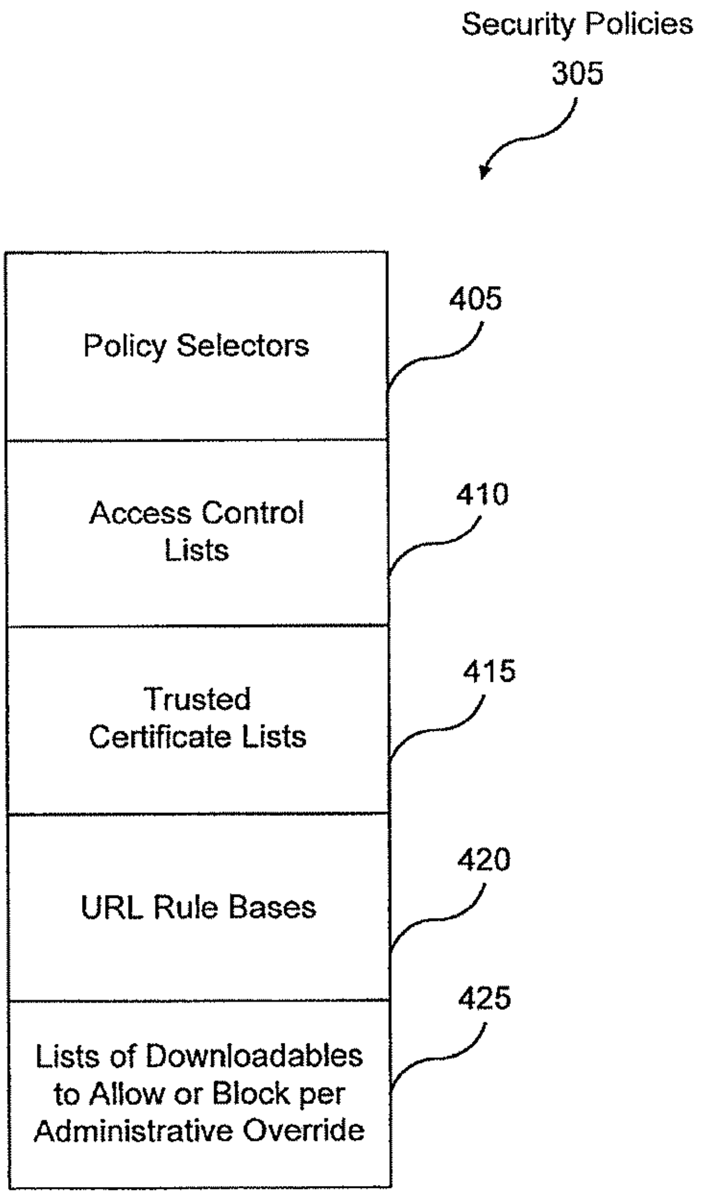

Security Policies
305
Policy Selectors
405
Access Control Lists
410
Trusted Certificate Lists
415
URL Rule Bases
420
Lists of Downloadables to Allow or Block per Administrative Override
425

FIG. 4

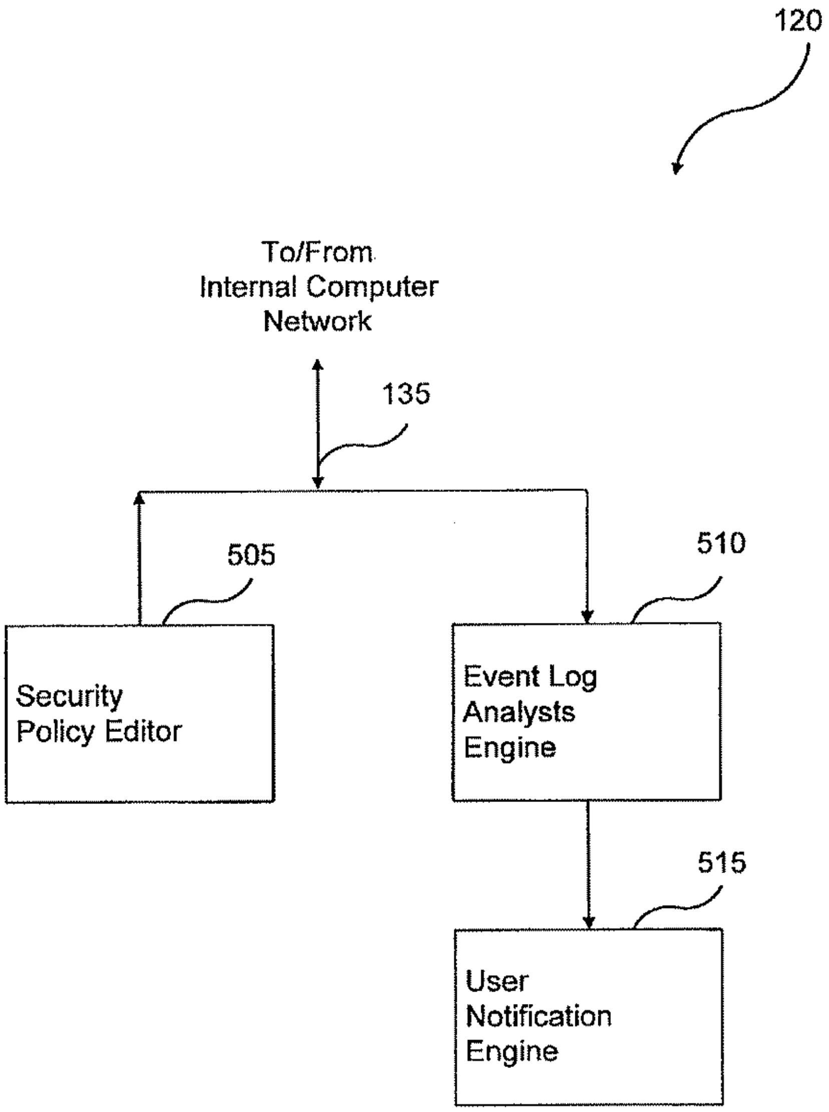

120
To/From
Internal Computer
Network
135
505
510
Security
Policy Editor
Event Log
Analysts
Engine
515
User
Notification
Engine

FIG. 5

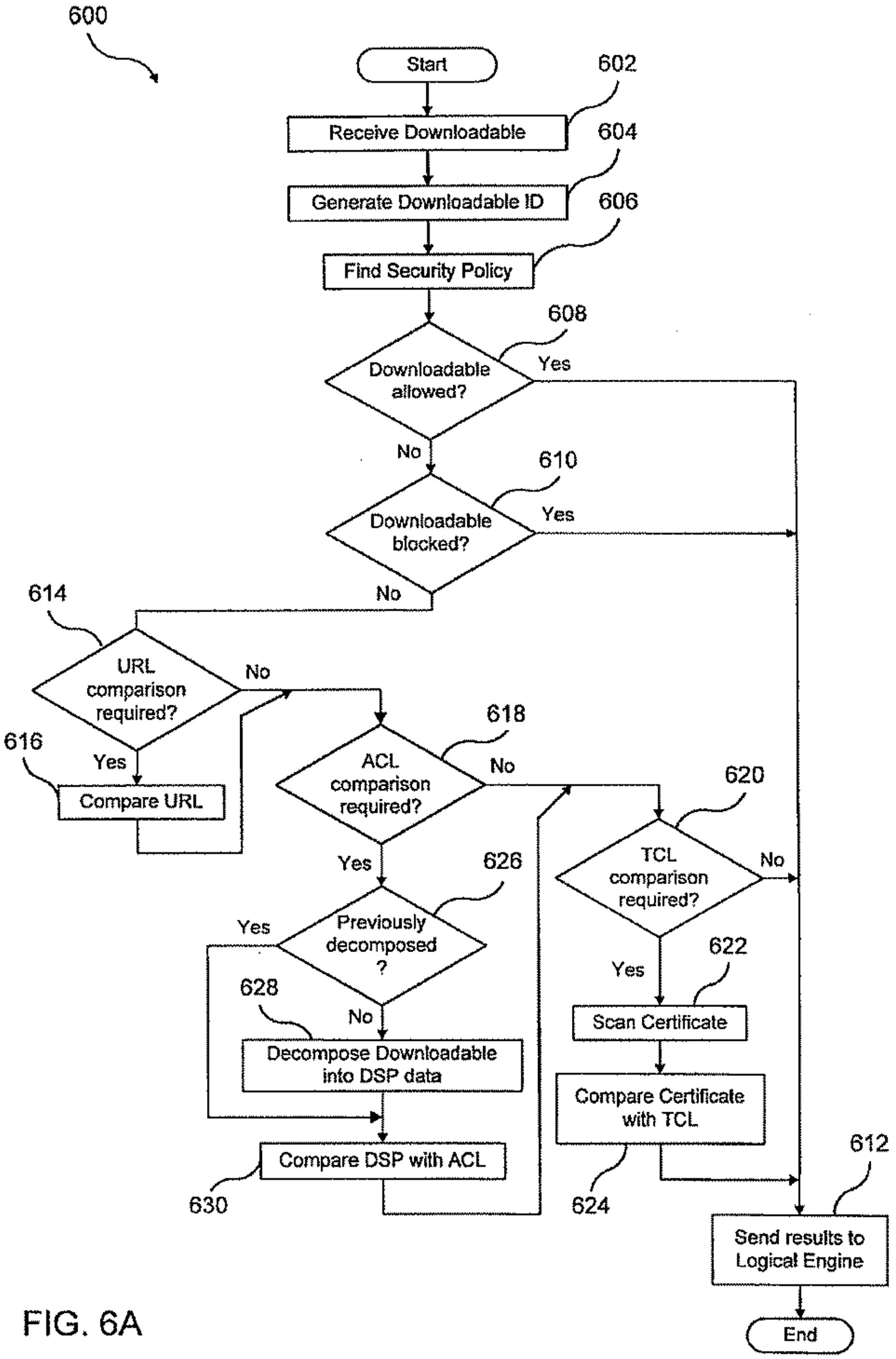

600
Start
602
Receive Downloadable
604
Generate Downloadable ID
606
Find Security Policy
608
Downloadable allowed?
Yes
No
610
Downloadable blocked?
Yes
No
614
URL comparison required?
No
616
Yes
Compare URL
618
ACL comparison required?
No
Yes
620
TCL comparison required?
No
626
Previously decomposed ?
Yes
628
No
Decompose Downloadable into DSP data
Compare DSP with ACL
630
Yes
622
Scan Certificate
Compare Certificate with TCL
624
612
Send results to Logical Engine
End

FIG. 6A

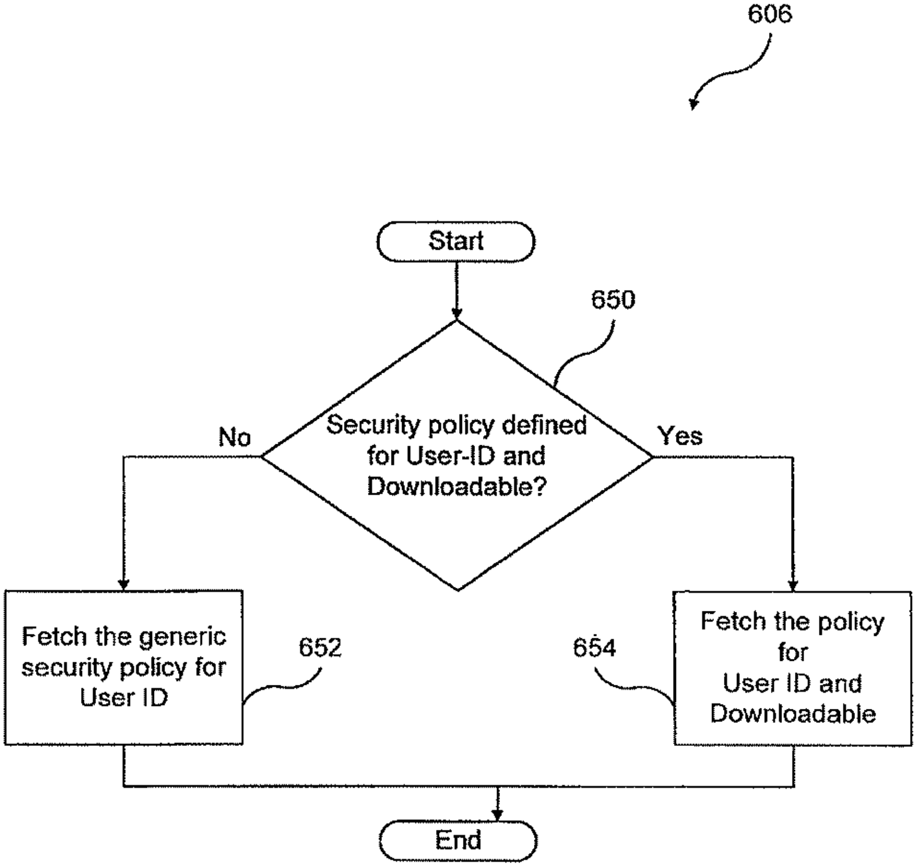

606
Start
650
Security policy defined for User-ID and Downloadable?
No
Yes
Fetch the generic security policy for User ID
652
654
Fetch the policy for User ID and Downloadable
End

FIG. 6B

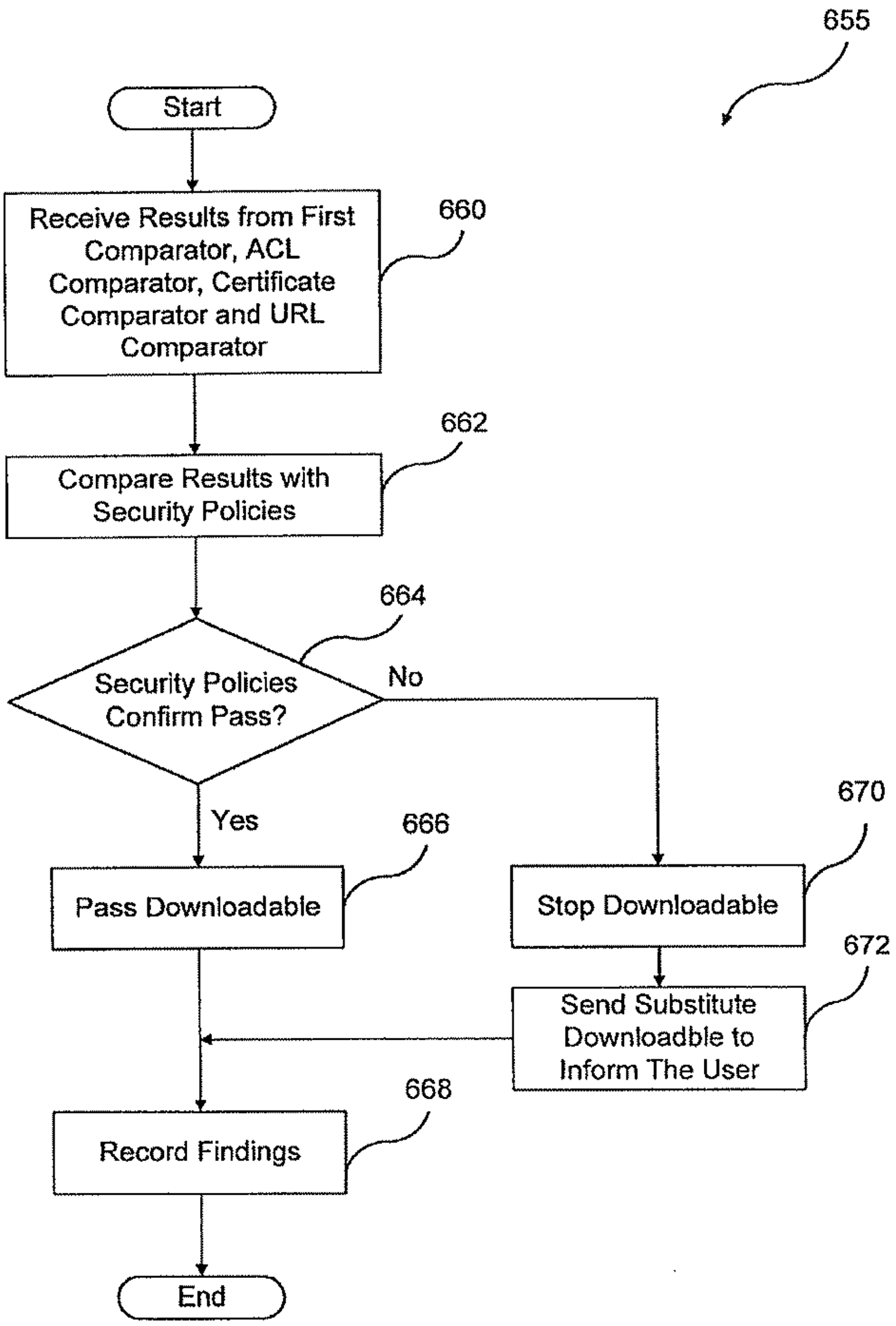

655
Start
Receive Results from First Comparator, ACL Comparator, Certificate Comparator and URL Comparator
660
Compare Results with Security Policies
662
Security Policies Confirm Pass?
664
No
Yes
Pass Downloadable
666
Stop Downloadable
670
Send Substitute Downloadble to Inform The User
672
Record Findings
668
End

FIG. 6C

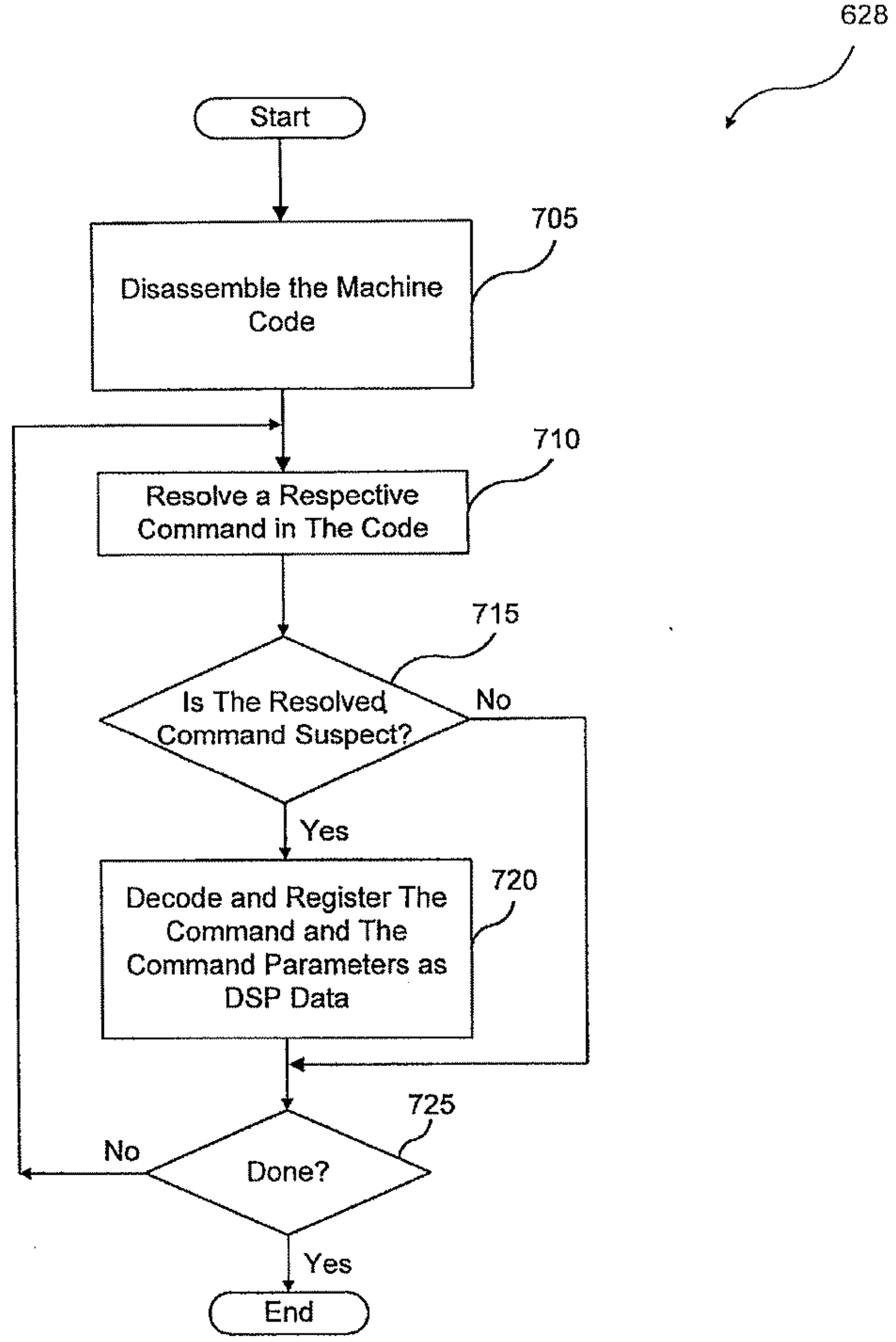

628
Start
705
Disassemble the Machine Code
710
Resolve a Respective Command in The Code
715
Is The Resolved Command Suspect?
No
Yes
720
Decode and Register The Command and The Command Parameters as DSP Data
725
No
Done?
Yes
End

FIG. 7

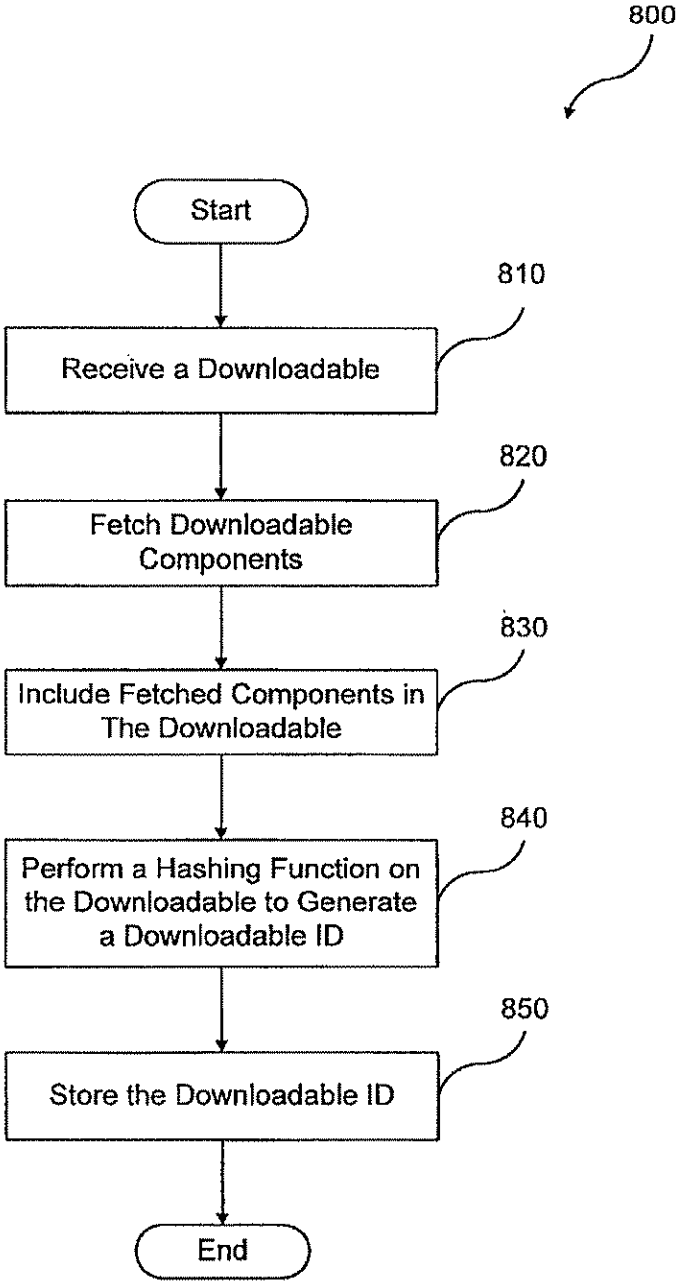

800
Start
810
Receive a Downloadable
820
Fetch Downloadable Components
830
Include Fetched Components in The Downloadable
840
Perform a Hashing Function on the Downloadable to Generate a Downloadable ID
850
Store the Downloadable ID
End

FIG. 8

SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES

INCORPORATION BY REFERENCE TO RELATED APPLICATION

This application hereby incorporates by reference related U.S. patent application Ser. No. 08/790,097, entitled "System and Method for Protecting a Client from Hostile Downloadables," filed on Jan. 29, 1997, by inventor Shlomo Touboul.

PRIORITY REFERENCE TO PROVISIONAL APPLICATION

This application claims benefit of and hereby incorporates by reference provisional application Ser. No. 60/030,639, entitled "System and Method for Protecting a Computer from Hostile Downloadables," filed on Nov. 8, 1996, by inventor Shlomo Touboul.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and method for protecting a computer and a network from hostile Downloadables.

2. Description of the Background Art

The Internet is currently a collection of over 100,000 individual computer networks owned by governments, universities, nonprofit groups and companies, and is expanding at an accelerating rate. Because the Internet is public, the Internet has become a major source of many system damaging and system fatal application programs, commonly referred to as "viruses."

Accordingly, programmers continue to design computer and computer network security systems for blocking these viruses from attacking both individual and network computers. On the most part, these security systems have been relatively successful. However, these security systems are not configured to recognize computer viruses which have been attached to or configured as Downloadable application programs, commonly referred to as "Downloadables." A Downloadable is an executable application program, which is downloaded from a source computer and run on the destination computer. Downloadable is typically requested by an ongoing process such as by an Internet browser or web engine. Examples of Downloadables include Java™ applets designed for use in the Java™ distributing environment developed by Sun Microsystems, Inc., JavaScript scripts also developed by Sun Microsystems, Inc., ActiveX™ controls designed for use in the ActiveX™ distributing environment developed by the Microsoft Corporation, and Visual Basic also developed by the Microsoft Corporation. Therefore, a system and method are needed to protect a network from hostile Downloadables.

SUMMARY OF THE INVENTION

The present invention provides a system for protecting a network from suspicious Downloadables. The system comprises a security policy, an interface for receiving a Downloadable, and a comparator, coupled to the interface, for applying the security policy to the Downloadable to determine if the security policy has been violated. The Downloadable may include a Java™ applet, an ActiveX™ control, a JavaScript™ script, or a Visual Basic script. The

security policy may include a default security policy to be applied regardless of the client to whom the Downloadable is addressed, a specific security policy to be applied based on the client or the group to which the client belongs, or a specific policy to be applied based on the client/group and on the particular Downloadable received. The system uses an ID generator to compute a Downloadable ID identifying the Downloadable, preferably, by fetching all components of the Downloadable and performing a hashing function on the Downloadable including the fetched components.

Further, the security policy may indicate several tests to perform, including (1) a comparison with known hostile and non-hostile Downloadables; (2) a comparison with Downloadables to be blocked or allowed per administrative override; (3) a comparison of the Downloadable security profile data against access control lists; (4) a comparison of a certificate embodied in the Downloadable against trusted certificates; and (5) a comparison of the URL from which the Downloadable originated against trusted and untrusted URLs. Based on these tests, a logical engine can determine whether to allow or block the Downloadable.

The present invention further provides a method for protecting a computer from suspicious Downloadables. The method comprises the steps of receiving a Downloadable, comparing the Downloadable against a security policy to determine if the security policy has been violated, and discarding the Downloadable if the security policy has been violated.

It will be appreciated that the system and method of the present invention may provide computer protection from known hostile Downloadables. The system and method of the present invention may identify Downloadables that perform operations deemed suspicious. The system and method of the present invention may examine the Downloadable code to determine whether the code contains any suspicious operations, and thus may allow or block the Downloadable accordingly.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating a network system, in accordance with the present invention;

FIG. 2 is a block diagram illustrating details of the internal network security system of FIG. 1;

FIG. 3 is a block diagram illustrating details of the security program and the security database of FIG. 2;

FIG. 4 is a block diagram illustrating details of the security policies of FIG. 3;

FIG. 5 is a block diagram illustrating details of the security management console of FIG. 1;

FIG. 6A is a flowchart illustrating a method of examining for suspicious Downloadables, in accordance with the present invention;

FIG. 6B is a flowchart illustrating details of the step for finding the appropriate security policy of FIG. 6A;

FIG. 6C is a flowchart illustrating a method for determining whether an incoming Downloadable is to be deemed suspicious;

FIG. 7 is a flowchart illustrating details of the FIG. 6 step of decomposing a Downloadable; and

FIG. 8 is a flowchart illustrating a method 800 for generating a Downloadable ID for identifying a Downloadable.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 is a block diagram illustrating a network system 100, in accordance with the present invention. The network system **100** includes an external computer network **105**, such as the Wide Area Network (WAN) commonly referred to as the Internet, coupled via a communications channel **125** to an internal network security system **110**. The network system **100** further includes an internal computer network **115**, such as a corporate Local Area Network (LAN), coupled via a communications channel **130** to the internal network computer system **110** and coupled via a communications channel **135** to a security management console **120**.

The internal network security system **110** examines Downloadables received from external computer network **105**, and prevents Downloadables deemed suspicious from reaching the internal computer network **115**. It will be further appreciated that a Downloadable is deemed suspicious if it performs or may perform any undesirable operation, or if it threatens or may threaten the integrity of an internal computer network **115** component. It is to be understood that the term "suspicious" includes hostile, potentially hostile, undesirable, potentially undesirable, etc. Security management console **120** enables viewing, modification and configuration of the internal network security system **110**.

FIG. **2** is a block diagram illustrating details of the internal network security system **110**, which includes a Central Processing Unit (CPU) **205**, such as an Intel Pentium® microprocessor or a Motorola Power PC® microprocessor, coupled to a signal bus **220**. The internal network security system **110** further includes an external communications interface **210** coupled between the communications channel **125** and the signal bus **220** for receiving Downloadables from external computer network **105**, and an internal communications interface **225** coupled between the signal bus **220** and the communications channel **130** for forwarding Downloadables not deemed suspicious to the internal computer network **115**. The external communications interface **210** and the internal communications interface **225** may be functional components of an integral communications interface (not shown) for both receiving Downloadables from the external computer network **105** and forwarding Downloadables to the internal computer network **115**.

Internal network security system **110** further includes Input/Output (I/O) interfaces **215** (such as a keyboard, mouse and Cathode Ray Tube (CRT) display), a data storage device **230** such as a magnetic disk, and a Random-Access Memory (RAM) **235**, each coupled to the signal bus **220**. The data storage device **230** stores a security database **240**, which includes security information for determining whether a received Downloadable is to be deemed suspicious. The data storage device **230** further stores a users list **260** identifying the users within the internal computer network **115** who may receive Downloadables, and an event log **245** which includes determination results for each Downloadable examined and runtime indications of the internal network security system **110**. An operating system **250** controls processing by CPU **205**, and is typically stored in data storage device **230** and loaded into RAM **235** (as illustrated) for execution. A security program **255** controls examination of incoming Downloadables, and also may be stored in data storage device **230** and loaded into RAM **235** (as illustrated) for execution by CPU **205**.

FIG. **3** is a block diagram illustrating details of the security program **255** and the security database **240**. The security program **255** includes an ID generator **315**, a policy finder **317** coupled to the ID generator **315**, and a first comparator **320** coupled to the policy finder **317**. The first comparator **320** is coupled to a logical engine **333** via four separate paths, namely, via Path 1, via Path 2, via Path 3 and via Path 4. Path 1 includes a direct connection from the first comparator **320** to the logical engine **333**. Path 2 includes a code scanner coupled to the first comparator **320**, and an Access Control List (ACL) comparator **330** coupling the code scanner **325** to the logical engine **333**. Path 3 includes a certificate scanner **340** coupled to the first comparator **320**, and a certificate comparator **345** coupling the certificate scanner **340** to the logical engine **333**. Path 4 includes a Uniform Resource Locator (URL) comparator **350** coupling the first comparator **320** to the logical engine **3330**. A record-keeping engine **335** is coupled between the logical engine **333** and the event log **245**.

The security program **255** operates in conjunction with the security database **240**, which includes security policies **305**, known Downloadables **307**, known Certificates **309** and Downloadable Security Profile (DSP) data **310** corresponding to the known Downloadables **307**. Security policies **305** includes policies specific to particular users **260** and default (or generic) policies for determining whether to allow or block an incoming Downloadable. These security policies **305** may identify specific Downloadables to block, specific Downloadables to allow, or necessary criteria for allowing an unknown Downloadable. Referring to FIG. **4**, security policies **305** include policy selectors **405**, access control lists **410**, trusted certificate lists **415**, URL rule bases **420**, and lists **425** of Downloadables to allow or to block per administrative override.

Known Downloadables **307** include lists of Downloadables which Original Equipment Manufacturers (OEMs) know to be hostile, of Downloadables which OEMs know to be non-hostile, and of Downloadables previously received by this security program **255**. DSP data **310** includes the list of all potentially hostile or suspicious computer operations that may be attempted by each known Downloadable **307**, and may also include the respective arguments of these operations. An identified argument of an operation is referred to as "resolved." An unidentified argument is referred to as "unresolved." DSP data **310** is described below with reference to the code scanner **325**.

The ID generator **315** receives a Downloadable (including the URL from which it came and the userID of the intended recipient) from the external computer network **105** via the external communications interface **210**, and generates a Downloadable ID for identifying each Downloadable. The Downloadable ID preferably includes a digital hash of the complete Downloadable code. The ID generator **315** preferably prefetches all components embodied in or identified by the code for Downloadable ID generation. For example, the ID generator **315** may prefetch all classes embodied in or identified by the Java™ applet bytecode to generate the Downloadable ID. Similarly, the ID generator **315** may retrieve all components listed in the .INF file for an ActiveX™ control to compute a Downloadable ID. Accordingly, the Downloadable ID for the Downloadable will be the same each time the ID generator **315** receives the same Downloadable. The ID generator **315** adds the generated Downloadable ID to the list of known Downloadables **307** (if it is not already listed). The ID generator **315** then forwards the Downloadable and Downloadable ID to the policy finder **317**.

The policy finder **317** uses the userID of the intended user and the Downloadable ID to select the specific security policy **305** that shall be applied on the received Downloadable. If there is a specific policy **305** that was defined for the user (or for one of its super groups) and the Downloadable, then the policy is selected. Otherwise the generic policy **305** that was defined for the user (or for one of its super groups) is selected. The policy finder **317** then sends the policy to the first comparator **320**.

The first comparator **320** receives the Downloadable, the Downloadable ID and the security policy **305** from the policy finder **317**. The first comparator **320** examines the security policy **305** to determine which steps are needed for allowing the Downloadable. For example, the security policy **305** may indicate that, in order to allow this Downloadable, it must pass all four paths, Path 1, Path 2, Path 3 and Path 4. Alternatively, the security policy **305** may indicate that to allow the Downloadable, it must pass only one of the paths. The first comparator **320** responds by forwarding the proper information to the paths identified by the security policy **305**.

Path 1

In path 1, the first comparator **320** checks the policy selector **405** of the security policy **305** that was received from the policy finder **317**. If the policy selector **405** is either "Allowed" or "Blocked," then the first comparator **320** forwards this result directly to the logical engine **333**. Otherwise, the first comparator **320** invokes the comparisons in path 2 and/or path 3 and/or path 4 based on the contents of policy selector **405**. It will be appreciated that the first comparator **320** itself compares the Downloadable ID against the lists of Downloadables to allow or block per administrative override **425**. That is, the system security administrator can define specific Downloadables as "Allowed" or "Blocked."

Alternatively, the logical engine **333** may receive the results of each of the paths and based on the policy selector **405** may institute the final determination whether to allow or block the Downloadable. The first comparator **320** informs the logical engine **333** of the results of its comparison.

Path 2

In path 2, the first comparator **320** delivers the Downloadable, the Downloadable ID and the security policy **305** to the code scanner **325**. If the DSP data **310** of the received Downloadable is known, the code scanner **325** retrieves and forwards the information to the ACL comparator **330**. Otherwise, the code scanner **325** resolves the DSP data **310**. That is, the code scanner **325** uses conventional parsing techniques to decompose the code (including all prefetched components) of the Downloadable into the DSP data **310**. DSP data **310** includes the list of all potentially hostile or suspicious computer operations that may be attempted by a specific Downloadable **307**, and may also include the respective arguments of these operations. For example, DSP data **310** may include a READ from a specific file, a SEND to an unresolved host, etc. The code scanner **325** may generate the DSP data **310** as a list of all operations in the Downloadable code which could ever be deemed potentially hostile and a list of all files to be accessed by the Downloadable code. It will be appreciated that the code scanner **325** may search the code for any pattern, which is undesirable or suggests that the code was written by a hacker.

An Example List of Operations Deemed Potentially Hostile

File operations: READ a file, WRITE a file;

Network operations: LISTEN on a socket, CONNECT to a socket, SEND data, RECEIVE data, VIEW INTRANET;

Registry operations: READ a registry item, WRITE a registry item;

Operating system operations: EXIT WINDOWS, EXIT BROWSER, START PROCESS/THREAD, KILL PROCESS/THREAD, CHANGE PROCESS/THREAD PRIORITY, DYNAMICALLY LOAD A CLASS/LIBRARY, etc.; and

Resource usage thresholds: memory, CPU, graphics, etc.

In the preferred embodiment, the code scanner **325** performs a full-content inspection. However, for improved speed but reduced security, the code scanner **325** may examine only a portion of the Downloadable such as the Downloadable header. The code scanner **325** then stores the DSP data into DSP data **310** (corresponding to its Downloadable ID), and sends the Downloadable, the DSP data to the ACL comparator **330** for comparison with the security policy **305**.

The ACL comparator **330** receives the Downloadable, the corresponding DSP data and the security policy **305** from the code scanner **325**, and compares the DSP data against the security policy **305**. That is, the ACL comparator **330** compares the DSP data of the received Downloadable against the access control lists **410** in the received security policy **305**. The access control list **410** contains criteria indicating whether to pass or fail the Downloadable. For example, an access control list may indicate that the Downloadable fails if the DSP data includes a WRITE command to a system file. The ACL comparator **330** sends its results to the logical engine **333**.

Path 3

In path 3, the certificate scanner **340** determines whether the received Downloadable was signed by a certificate authority, such as VeriSign, Inc., and scans for a certificate embodied in the Downloadable. The certificate scanner **340** forwards the found certificate to the certificate comparator **345**. The certificate comparator **345** retrieves known certificates **309** that were deemed trustworthy by the security administrator and compares the found certificate with the known certificates **309** to determine whether the Downloadable was signed by a trusted certificate. The certificate comparator **345** sends the results to the logical engine **333**.

Path 4

In path 4, the URL comparator **350** examines the URL identifying the source of the Downloadable against URLs stored in the URL rule base **420** to determine whether the Downloadable comes from a trusted source. Based on the security policy **305**, the URL comparator **350** may deem the Downloadable suspicious if the Downloadable comes from an untrustworthy source or if the Downloadable did not come from a trusted source. For example, if the Downloadable comes from a known hacker, then the Downloadable may be deemed suspicious and presumed hostile. The URL comparator **350** sends its results to the logical engine **333**.

The logical engine **333** examines the results of each of the paths and the policy selector **405** in the security policy **305** to determine whether to allow or block the Downloadable. The policy selector **405** includes a logical expression of the results received from each of the paths. For example, the logical engine **333** may block a Downloadable if it fails any one of the paths, i.e., if the Downloadable is known hostile (Path 1), if the Downloadable may request suspicious operations (Path 2), if the Downloadable was not signed by a trusted certificate authority (Path 3), or if the Downloadable came from an untrustworthy source (Path 4). The logical engine **333** may apply other logical expressions according to the policy selector **405** embodied in the security policy **305**. If the policy selector **405** indicates that the Downloadable may pass, then the logical engine **333** passes the Downloadable to its intended recipient. Otherwise, if the policy selector **405** indicates that the Downloadable should be blocked, then the logical engine **333** forwards a non-hostile Downloadable to the intended recipient to inform the user that internal network security system **110** discarded the original Downloadable. Further, the logical engine **333** forwards a status report to the record-keeping engine **335**, which stores the reports in event log **245** in the data storage device **230** for subsequent review, for example, by the MIS director.

FIG. **5** is a block diagram illustrating details of the security management console **120**, which includes a security policy editor **505** coupled to the communications channel **135**, an event log analysis engine **510** coupled between communications channel **135** and a user notification engine **515**, and a Downloadable database review engine **520** coupled to the communications channel **135**. The security management console **120** further includes computer components similar to the computer components illustrated in FIG. **2**.

The security policy editor **505** uses an I/O interface similar to I/O interface **215** for enabling authorized user modification of the security policies **305**. That is, the security policy editor **505** enables the authorized user to modify specific security policies **305** corresponding to the users **260**, the default or generic security policy **305**, the Downloadables to block per administrative override, the Downloadables to allow per administrative override, the trusted certificate lists **415**, the policy selectors **405**, the access control lists **410**, the URLs in the URL rule bases **420**, etc. For example, if the authorized user learns of a new hostile Downloadable, then the user can add the Downloadable to the Downloadables to block per system override.

The event log analysis engine **510** examines the status reports contained in the event log **245** stored in the data storage device **230**. The event log analysis engine **510** determines whether notification of the user (e.g., the security system manager or MIS director) is warranted. For example, the event log analysis engine **510** may warrant user notification whenever ten (10) suspicious Downloadables have been discarded by internal network security system **110** within a thirty (30) minute period, thereby flagging a potential imminent security threat. Accordingly, the event log analysis engine **510** instructs the user notification engine **515** to inform the user. The user notification engine **515** may send an e-mail via internal communications interface **220** or via external communications interface **210** to the user, or may display a message on the user's display device (not shown).

FIG. **6A** is a flowchart illustrating a method **600** for protecting an internal computer network **115** from suspicious Downloadables. Method **600** begins with the ID generator **315** in step **602** receiving a Downloadable. The ID generator **315** in step **604** generates a Downloadable ID identifying the received Downloadable, preferably, by generating a digital hash of the Downloadable code (including prefetched components). The policy finder **317** in step **606** finds the appropriate security policy **305** corresponding to the userID specifying intended recipient (or the group to which the intended recipient belongs) and the Downloadable. The selected security policy **305** may be the default security policy **305**. Step **606** is described in greater detail below with reference to FIG. 6B.

The first comparator **320** in step **608** examines the lists of Downloadables to allow or to block per administrative override **425** against the Downloadable ID of the incoming Downloadable to determine whether to allow the Downloadable automatically. If so, then in step **612** the first comparator **320** sends the results to the logical engine **333**. If not, then the method **600** proceeds to step **610**. In step **610**, the first comparator **620** examines the lists of Downloadables to block per administrative override **425** against the Downloadable ID of the incoming Downloadable for determining whether to block the Downloadable automatically. If so, then the first comparator **420** in step **612** sends the results to the logical engine **333**. Otherwise, method **600** proceeds to step **614**.

In step **614**, the first comparator **320** determines whether the security policy **305** indicates that the Downloadable should be tested according to Path 4. If not, then method **600** jumps to step **618**. If so, then the URL comparator **350** in step **616** compares the URL embodied in the incoming Downloadable against the URLs of the URL rules bases **420**, and then method **600** proceeds to step **618**.

In step **618**, the first comparator **320** determines whether the security policy **305** indicates that the Downloadable should be tested according to Path 2. If not, then method **600** jumps to step **620**. Otherwise, the code scanner **235** in step **626** examines the DSP data **310** based on the Downloadable ID of the incoming Downloadable to determine whether the Downloadable has been previously decomposed. If so, then method **600** jumps to step **630**. Otherwise, the code scanner **325** in step **628** decomposes the Downloadable into DSP data. Downloadable decomposition is described in greater detail with reference to FIG. **7**. In step **630**, the ACL comparator **330** compares the DSP data of the incoming Downloadable against the access control lists **410** (which include the criteria necessary for the Downloadable to fail or pass the test).

In step **620**, the first comparator **320** determines whether the security policy **305** indicates that the Downloadable should be tested according to Path 3. If not, then method **600** returns to step **612** to send the results of each of the test performed to the logical engine **333**. Otherwise, the certificate scanner **622** in step **622** scans the Downloadable for an embodied certificate. The certificate comparator **345** in step **624** retrieves trusted certificates from the trusted certificate lists (TCL) **415** and compares the embodied certificate with the trusted certificates to determine whether the Downloadable has been signed by a trusted source. Method **600** then proceeds to step **612** by the certificate scanner **345** sending the results of each of the paths taken to the logical engine **333**. The operations of the logical engine **333** are described in greater detail below with reference to FIG. **6C**. Method **600** then ends.

One skilled in the art will recognize that the tests may be performed in a different order, and that each of the tests need not be performed. Further, one skilled in the art will recognize that, although path 1 is described in FIG. **6A** as an automatic allowance or blocking, the results of Path 1 may be another predicate to be applied by the logical engine **333**. Further, although the tests are shown serially in FIG. **6A**, the tests may be performed in parallel as illustrated in FIG. **3**.

FIG. **6B** is a flowchart illustrating details of step **606** of FIG. **6A** (referred to herein as method **606**). Method **606** begins with the policy finder **317** in step **650** determining whether security policies **305** include a specific security policy corresponding to the userID and the Downloadable. If so, then the policy finder **317** in step **654** fetches the corresponding specific policy **305**. If not, then the policy finder **317** in step **652** fetches the default or generic security policy **305** corresponding to the userID. Method **606** then ends.

FIG. **6C** is a flowchart illustrating details of a method **655** for determining whether to allow or to block the incoming Downloadable. Method **655** begins with the logical engine **333** in step **660** receiving the results from the first comparator **320**, from the ACL comparator **330**, from the certificate comparator **345** and from the URL comparator **350**. The logical engine **333** in step **662** compares the results with the policy selector **405** embodied in the security policy **305**, and in step **664** determines whether the policy selector **405** confirms the pass. For example, the policy selector **405** may indicate that the logical engine **333** pass the Downloadable if it passes one of the tests of Path 1, Path 2, Path 3 and Path 4. If the policy selector **405** indicates that the Downloadable should pass, then the logical engine **333** in step **666** passes the Downloadable to the intended recipient. In step **668**, the logical engine **333** sends the results to the record-keeping engine **335**, which in turn stores the results in the event log **245** for future review. Method **655** then ends. Otherwise, if the policy selector **405** in step **664** indicates that the Downloadable should not pass, then the logical engine **333** in step **670** stops the Downloadable and in step **672** sends a non-hostile substitute Downloadable to inform the user that the incoming Downloadable has been blocked. Method **655** then jumps to step **668**.

FIG. **7** is a flowchart illustrating details of step **628** of FIG. **6**A (referred to herein as method **628**) for decomposing a Downloadable into DSP data **310**. Method **628** begins in step **705** with the code scanner **325** disassembling the machine code of the Downloadable. The code scanner **325** in step **710** resolves a respective command in the machine code, and in step **715** determines whether the resolved command is suspicious (e.g., whether the command is one of the operations identified in the list described above with reference to FIG. **3**). If not, then the code scanner **325** in step **725** determines whether it has completed decomposition of the Downloadable, i.e., whether all operations in the Downloadable code have been resolved. If so, then method **628** ends. Otherwise, method **628** returns to step **710**.

Otherwise, if the code scanner **325** in step **715** determines that the resolved command is suspect, then the code scanner **325** in step **720** decodes and registers the suspicious command and its command parameters as DSP data **310**. The code scanner **325** in step **720** registers the commands and command parameters into a format based on command class (e.g., file operations, network operations, registry operations, operating system operations, resource usage thresholds). Method **628** then jumps to step **725**.

FIG. **8** is a flowchart illustrating a method **800** for generating a Downloadable ID for identifying a Downloadable. Method **800** begins with the ID generator **315** in step **810** receiving a Downloadable from the external computer network **105**. The ID generator **315** in step **820** may fetch some or all components referenced in the Downloadable code, and in step **830** includes the fetched components in the Downloadable code. The ID generator **315** in step **840** performs a hashing function on at least a portion of the Downloadable code to generate a Downloadable ID. The ID generator **315** in step **850** stores the generated Downloadable ID in the security database **240** as a reference to the DSP data **310**. Accordingly, the Downloadable ID will be the same for the identical Downloadable each time it is encountered.

The foregoing description of the preferred embodiments of the invention is by way of example only, and other variations of the above-described embodiments and methods are provided by the present invention. For example, although the invention has been described in a system for protecting an internal computer network, the invention can be embodied in a system for protecting an individual computer. Components of this invention may be implemented using a programmed general purpose digital computer, using application specific integrated circuits, or using a network of interconnected conventional components and circuits. The embodiments described herein have been presented for purposes of illustration and are not intended to be exhaustive or limiting. Many variations and modifications are possible in light of the foregoing teaching. The system is limited only by the following claims.

What is claimed is:

1. A computer-based method, comprising the steps of:

receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;

comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and

preventing execution of the Downloadable by the client if the security policy has been violated.

2. The method of claim **1**, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.

3. The method of claim **2**, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.

4. The method of claim **1**, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.

5. The method of claim **1**, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.

6. The method of claim **5**, wherein the known URL is a trusted URL.

7. The method of claim **5**, wherein the known URL is an untrusted URL.

8. The method of claim **1**, wherein the Downloadable includes a Java™ applet.

9. The method of claim **1**, wherein the Downloadable includes an ActiveX™ control.

10. The method of claim **1**, wherein the Downloadable includes a JavaScript™ script.

11. The method of claim **1**, wherein the Downloadable includes a Visual Basic script.

12. The method of claim **1**, wherein

the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.

13. The method of claim **1**, wherein

the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.

14. The method of claim **1**, wherein

the client belongs to a particular group; and

the security policy includes a specific security policy corresponding to the particular group.

15. The method of claim **1**, further comprising, after preventing execution of the Downloadable, the step of sending a substitute non-hostile Downloadable to the client for informing the client.

16. The method of claim **1**, further comprising, after preventing execution of the Downloadable, the step of recording the violation in an event log.

17. The method of claim **1**, further comprising the step of computing a Downloadable ID to identify the Downloadable.

18. The method of claim **16**, further comprising the steps of fetching components identified by the Downloadable and including the fetched components in the Downloadable.

19. The method of claim 18, further comprising the step of performing a hashing function on the Downloadable to compute a Downloadable ID to identify the Downloadable.

20. The method of claim 18, further comprising the step of fetching all components identified by the Downloadable.

21. The method of claim 1 further comprising the step of examining the intended recipient userID to determine the appropriate security policy.

22. The method of claim 20, wherein the appropriate security policy includes a default security policy.

23. The method of claim 1, further comprising the step of examining the Downloadable to determine the appropriate security policy.

24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.

25. The method of claim 24, wherein the known Downloadable is hostile.

26. The method of claim 24, wherein the known Downloadable is non-hostile.

27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.

28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.

29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.

30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.

31. The method of claim 1, further comprising the steps of recognizing the incoming Downloadable, and obtaining the Downloadable security profile data for the incoming Downloadable from memory.

32. A system for execution by a server that serves as a gateway to a client, the system comprising:

a security policy;

an interface for receiving an incoming Downloadable addressed to a client;

a comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and

a logical engine for preventing execution of the Downloadable by the client if the security policy has been violated.

33. The system of claim 32, wherein the Downloadable includes a Java™ applet.

34. The system of claim 32, wherein the Downloadable includes ActiveX™ control.

35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.

36. The system of claim 32, wherein the Downloadable includes a Visual Basic script.

37. The system of claim 32, wherein

the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.

38. The system of claim 32, wherein

the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.

39. The system of claim 32, wherein

the client belongs to a particular group; and

the security policy includes a specific security policy corresponding to the particular group.

40. The system of claim 32, further comprising an ID generator coupled to the interface for computing a Downloadable ID identifying the Downloadable.

41. The system of claim 40, wherein the ID generator prefetches all components of the Downloadable and uses all components to compute the Downloadable ID.

42. The system of claim 41, wherein the ID generator computes the digital hash of all the prefetched components.

43. The system of claim 32, further comprising a policy finder for finding the security policy.

44. The system of claim 43, wherein the policy finder finds the security policy based on the user.

45. The system of claim 43 wherein the policy finder finds the security policy based on the user and the Downloadable.

46. The system of claim 43, wherein the policy finder obtains the default security policy.

47. The system of claim 32 wherein the comparator examines the security policy to determine which tests to apply.

48. The system of claim 47 wherein the comparator compares the Downloadable against a known Downloadable.

49. The system of claim 48, wherein the known Downloadable is hostile.

50. The system of claim 48, wherein the known Downloadable is non-hostile.

51. The system of claim 32, wherein the security policy identifies a Downloadable to be blocked per administrative override.

52. The system of claim 32, wherein the security policy identifies a Downloadable to be allowed per administrative override.

53. The system of claim 32, wherein

the comparator sends a substitute non-hostile Downloadable to the client for informing the client.

54. The system of claim 32, further comprising a code scanner coupled to the comparator for decomposing the Downloadable into the Downloadable security profile data.

55. The system of claim 54, further comprising an ACL comparator coupled to the code scanner for comparing the Downloadable security profile data against an access control list.

56. The system of claim 32, further comprising a certificate scanner coupled to the comparator for examining the Downloadable for a certificate.

57. The system of claim 56, further comprising a certificate comparator coupled to the certificate scanner for comparing the certificate against a trusted certificate.

58. The system of claim 32, further comprising a URL comparator coupled to the comparator for comparing the URL from which the Downloadable originated against a known URL.

59. The system of claim 58, wherein the known URL identifies an untrusted URL.

60. The system of claim 58, wherein the known URL identifies a trusted URL.

61. The system of claim 31, wherein the logical engine responds according to the security policy.

62. The system of claim 31, further comprising a recordkeeping engine coupled to the comparator for recording results in an event log.

63. The system of claim 32, further comprising memory storing the Downloadable security profile data for the incoming Downloadable.

64. A system for execution on a server that serves as a gateway to a client, comprising:

means for receiving an incoming Downloadable addressed to a client;

means for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and

means for preventing execution of the Downloadable by the client if the security policy has been violated.

65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:

receiving an incoming Downloadable addressed to a client;

comparing Downloadable security profile data pertaining to the Downloadable against a security policy to determine if the security policy has been violated; and

preventing execution of the Downloadable by the client if the security policy has been violated.

66. A method, comprising:

receiving a Downloadable;

decomposing the Downloadable into Downloadable security profile data; the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable,

comparing the Downloadable security profile data against a security policy; and

preventing execution of the Downloadable if the Downloadable security profile data violates the security policy.

67. The method of claim 66, further comprising:

fetching all components referenced by the Downloadable;

performing a hashing function of the Downloadable and the components fetched to compute a Downloadable ID; and

storing the Downloadable security profile data and the Downloadable ID in memory.

68. A method, comprising:

providing memory storing known-Downloadable security profile data and a that includes a list a suspicious computer operations that may be attempted by a Downloadable known-Downloadable ID corresponding to the Downloadable security profile data;

receiving an incoming Downloadable;

fetching all components referenced by the incoming Downloadable;

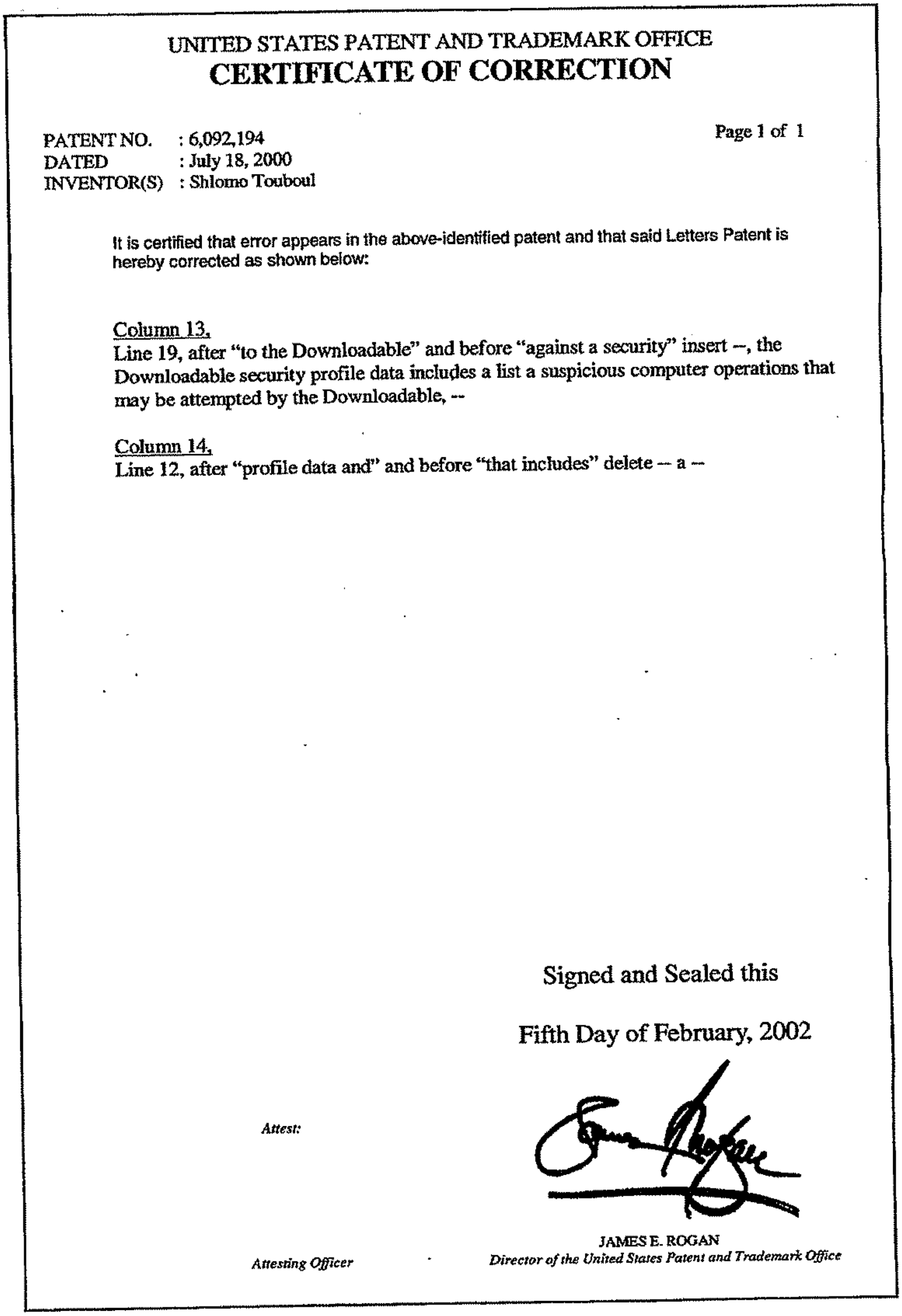
performing a hashing function of the Downloadable and the components to compute an incoming-Downloadable ID;

comparing the known-Downloadable ID against the incoming-Downloadable ID;

retrieving the Downloadable security profile data if the known-Downloadable ID and the incoming-Downloadable ID match; and

comparing the Downloadable security profile data against a security policy to determine if the incoming Downloadable violates the security policy.

* * * * *



UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

Page 1 of 1

PATENT NO. : 6,092,194
DATED : July 18, 2000
INVENTOR(S) : Shlomo Touboul

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 13,
Line 19, after "to the Downloadable" and before "against a security" insert --, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, --

Column 14,
Line 12, after "profile data and" and before "that includes" delete -- a --

Signed and Sealed this

Fifth Day of February, 2002

Attest:

JAMES E. ROGAN
Director of the United States Patent and Trademark Office

Attesting Officer

EXHIBIT B

US006804780B1

(12) **United States Patent**
Touboul

(10) **Patent No.: US 6,804,780 B1**
(45) **Date of Patent: *Oct. 12, 2004**

(54) **SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES**

(75) Inventor: **Shlomo Touboul**, Kefar-haim (IL)

(73) Assignee: **Finjan Software, Ltd.**, Netanya (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/539,667**

(22) Filed: **Mar. 30, 2000**

Related U.S. Application Data

(63) Continuation of application No. 08/964,388, filed on Nov. 6, 1997, now Pat. No. 6,092,194.

(60) Provisional application No. 60/030,639, filed on Nov. 8, 1996.

(51) **Int. Cl.**^{7} **H04L 9/00**; G06F 11/30

(52) **U.S. Cl.** **713/181**; 713/201; 713/176; 717/178

(58) **Field of Search** 713/200, 201, 713/176, 181; 709/223, 225, 227, 229; 717/168–178

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,077,677	A		12/1991	Murphy et al.	
5,359,659	A		10/1994	Rosenthal	
5,361,359	A		11/1994	Tajalli et al.	
5,485,409	A		1/1996	Gupta et al.	
5,485,575	A		1/1996	Chess et al.	
5,572,643	A		11/1996	Judson	
5,579,509	A	*	11/1996	Furtney et al.	.... 703/27
5,606,668	A		2/1997	Shwed	
5,623,600	A		4/1997	Ji et al.	
5,638,446	A		6/1997	Rubin	
5,692,047	A		11/1997	McManis	
5,692,124	A		11/1997	Holden et al.	
5,720,033	A		2/1998	Deo	
5,724,425	A		3/1998	Chang et al.	
5,740,248	A		4/1998	Fieres et al.	
5,761,421	A		6/1998	van Hoff et al.	

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	1091276	A1	*	4/2001	.... G06F/1/00
EP	1132796	A1	*	9/2001	.... G06F/1/00

OTHER PUBLICATIONS

Khare, "Microsoft Authenticode Analyzed" Jul. 22, 1996, xent.com/FoRK-archive/summer96/0338.html, p. 1–2.*

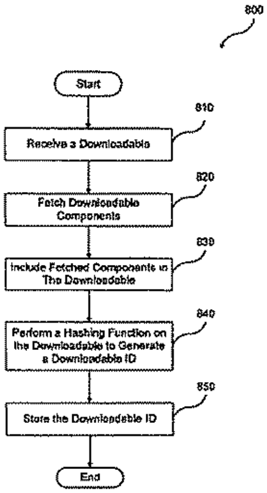
(List continued on next page.)

Primary Examiner—Ayaz Sheikh
Assistant Examiner—Christopher Revak
(74) *Attorney, Agent, or Firm*—Squire, Sanders & Dempsey, L.L.P.

(57) **ABSTRACT**

A computer-based method for generating a Downloadable ID to identify a Downloadable, including obtaining a Downloadable that includes one or more references to software components required by the Downloadable, fetching at least one software component identified by the one or more references, and performing a function on the Downloadable and the fetched software components to generate a Downloadable ID. A system and a computer-readable storage medium are also described and claimed.

18 Claims, 10 Drawing Sheets


800
Start
810
Receive a Downloadable
820
Fetch Downloadable Components
830
Include Fetched Components in The Downloadable
840
Perform a Hashing Function on the Downloadable to Generate a Downloadable ID
850
Store the Downloadable ID
End

U.S. PATENT DOCUMENTS

5,765,205	A		6/1998	Breslau et al.	
5,784,459	A		7/1998	Devarakonda et al.	
5,796,952	A		8/1998	Davis et al.	
5,805,829	A		9/1998	Cohen et al.	
5,832,208	A		11/1998	Chen et al.	
5,832,274	A	*	11/1998	Cutler et al.	717/171
5,850,559	A		12/1998	Angelo et al.	
5,859,966	A		1/1999	Hayman et al.	
5,864,683	A		1/1999	Boebert et al.	
5,892,904	A		4/1999	Atkinson et al.	
5,951,698	A		9/1999	Chen et al.	
5,956,481	A		9/1999	Walsh et al.	
5,974,549	A		10/1999	Golan	
5,978,484	A	*	11/1999	Apperson et al.	705/54
5,983,348	A		11/1999	Ji	
6,092,194	A	*	7/2000	Touboul	713/200
6,154,844	A	*	11/2000	Touboul et al.	713/201
6,339,829	B1	*	1/2002	Beadle et al.	713/201

OTHER PUBLICATIONS

"Release Notes for the Microsfot ActiveX Development Kit", Aug. 13, 1996, activex.adsp.or.jp/inetsdk/readme.txt, p. 1–10.*

"Microsoft ActiveX Software Development Kit" Aug. 12, 1996, activex.adsp.or.jp/inetsdk/help/overview.htm, p. 1–6.*

Doyle et al, "Microsoft Press Computer Dictionary" 1993, Microsoft Press, 2nd Edition, p. 137–138.*

Schmitt, ".EXE. files, OS–2 style" Nov. 1988, PC Tech Journal via dialog search, vol. 6, #11, p. 76–78.*

Jim K. Omura, "Novel Applications of Cryptography in Digital Communications", IEEE Communications Magazine, May, 1990; pp. 21–29.

Okamoto, E. et al., "ID–Based Authentication System For Computer Virus Detection", IEEE/IEE Electronic Library online, Electronics Letters, vol. 26, Issue 15, ISSN 0013/5194, Jul. 19, 1990, Abstract and pp. 1169–1170. URL: http://iel.ihs.com:80/cgi–bin/iel_cgi?se . . . 2ehts%26ViewTemplate%3ddocview%5fb%2ehts.

IBM AntiVirus User's Guide Version 2.4, International Business Machines Corporation, Nov. 15, 1995, pp. 6–7.

Norvin Leach et al, "IE 3.0 Applets Will Earn Certification", PC Week, vol. 13, No. 29, Jul. 22, 1996, 2 pages.

"Finjan Software Releases SurfinBoard, Industry's First JAVA Security Product For the World Wide Web", Article published on the Internet by Finjan Softwre Ltd., Jul. 29, 1996, 1 page.

"Powerful PC Security for the New World of Java™ and Downloadables, Surfin Shield™" Article published on the Internet by Finjan Software Ltd., 1996, 2 Pages.

Microsoft® Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet", Microsoft Corporation, Oct. 1996, including Abstract, Contents, Introduction and pp. 1–10.

"Finjan Announces a Personal Java™ Firewall For Web Browsers—the SurfinShield™ 1.6 (formerly known as SurfinBoard)", Press Release of Finjan Releases SurfinShield 1.6, Oct. 21, 1996, 2 pages.

Company Profile "Finjan—Safe Surfing, The Java Security Solutions Provider", Article published on the Internet by Finjan Software Ltd., Oct. 31, 1996, 3 pages.

"Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0" Las Vegas Convention Center/Pavilion 5 P5551, Nov. 18, 1996, 3 pages.

"Java Security: Issues & Solutions" Article published on the Internet by Finjan Software Ltd., 1996, 8 pages.

"Products" Article published on the Internet, 7 pages.

Mark LaDue, "Online Business Consulant: Java Security: Whose Business Is It?" Article published on the Internet, Home Page Press, Inc. 1996, 4 pages.

Web Page Article "Frequently Asked Questions About Authenticode", Microsoft Corporation, last updated Feb. 17, 1997, Printed Dec. 23, 1998. URL: http://www.microsoft.com/workshop/security/authcode/signfaq.asp#9, pp. 1–13.

Zhang, X.N., "Secure Code Distribution", IEEE/IEE Electronic Library online, Computer, vol. 30, Issue 6, Jun., 1997, pp. 76–79.

* cited by examiner

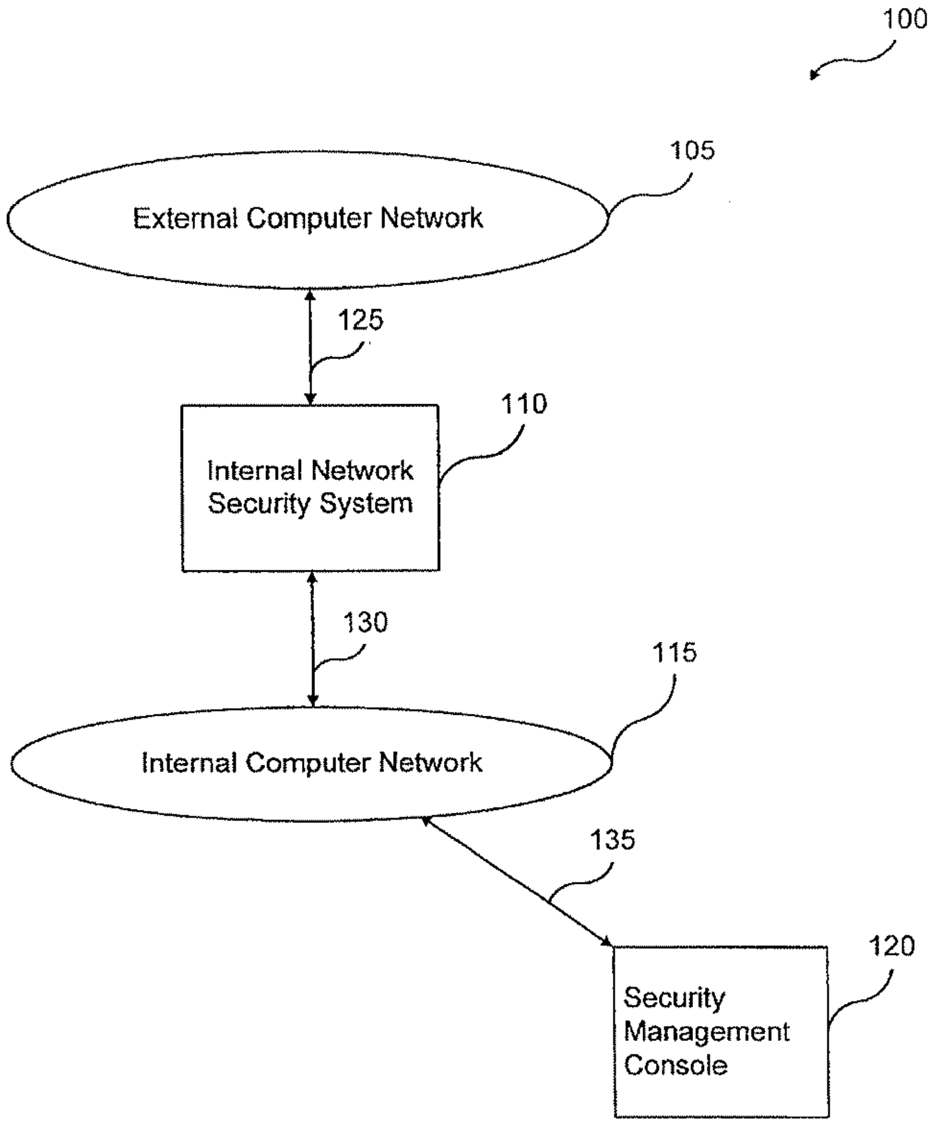

100
105
External Computer Network
125
110
Internal Network Security System
130
115
Internal Computer Network
135
120
Security Management Console

FIG. 1

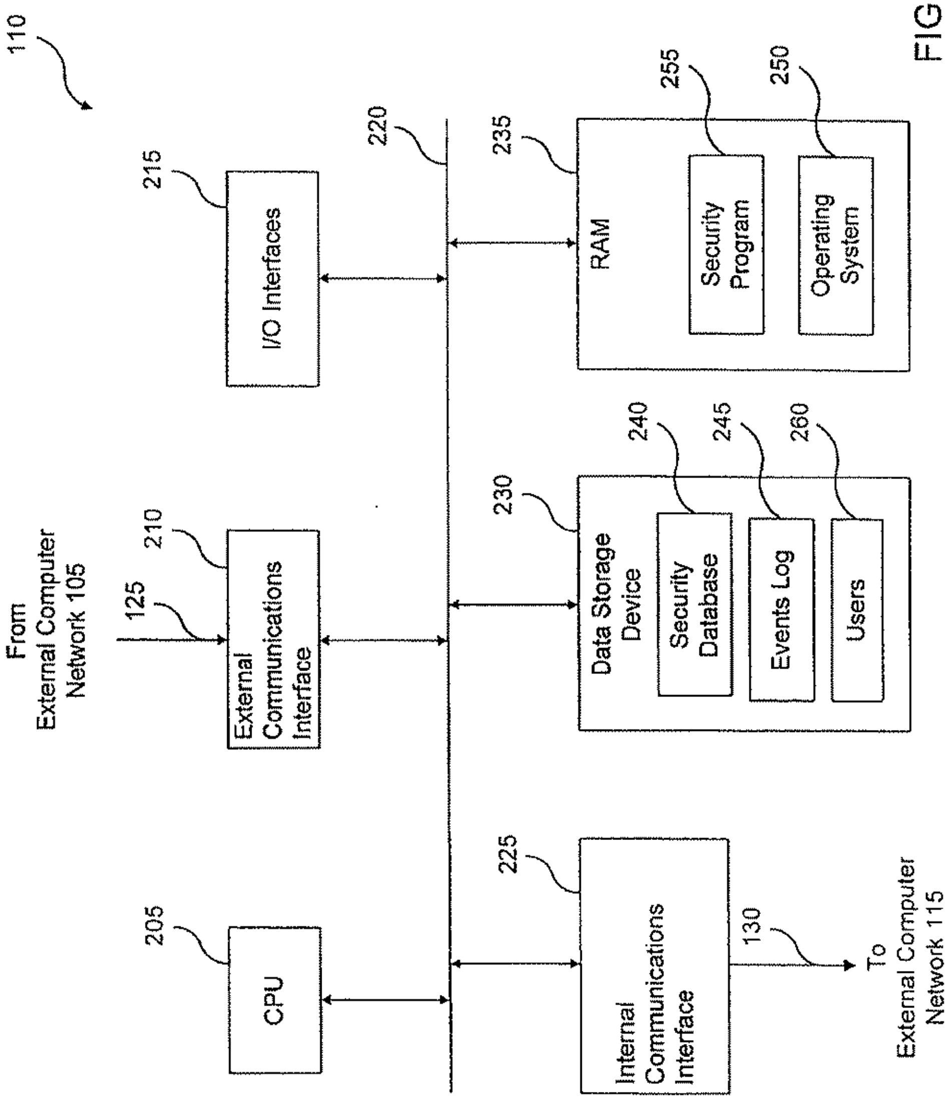

110
From
External Computer
Network 105
125
210
External
Communications
Interface
215
I/O Interfaces
205
CPU
220
235
RAM
255
Security
Program
250
Operating
System
230
Data Storage
Device
240
Security
Database
245
Events Log
260
Users
225
Internal
Communications
Interface
130
To
External Computer
Network 115

FIG. 2

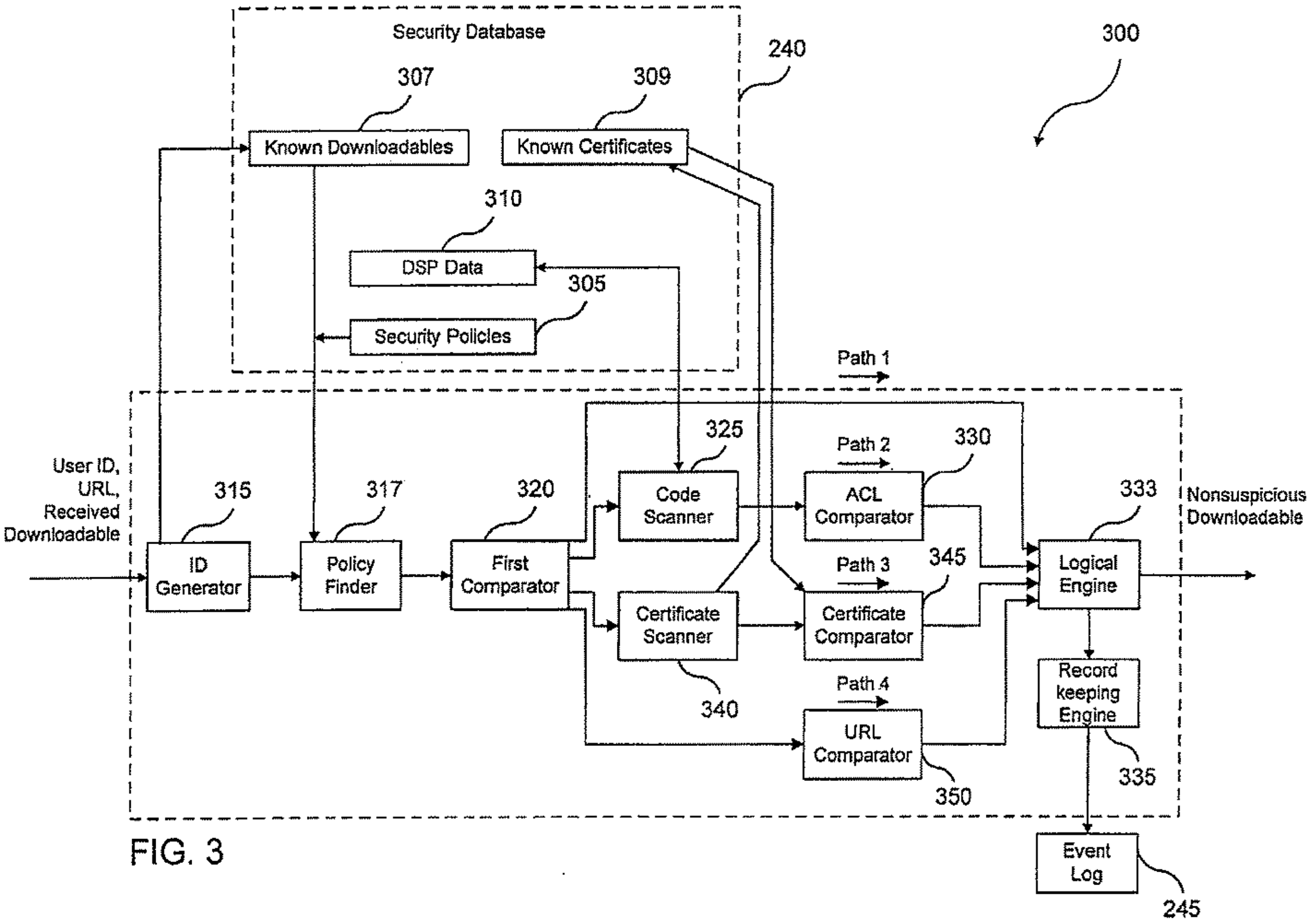

300
Security Database
240
307
Known Downloadables
309
Known Certificates
310
DSP Data
305
Security Policies
Path 1
User ID,
URL,
Received
Downloadable
315
ID
Generator
317
Policy
Finder
320
First
Comparator
325
Code
Scanner
Path 2
330
ACL
Comparator
Path 3
345
Certificate
Comparator
Certificate
Scanner
340
Path 4
URL
Comparator
350
333
Logical
Engine
Nonsuspicious
Downloadable
Record
keeping
Engine
335
Event
Log
245

FIG. 3

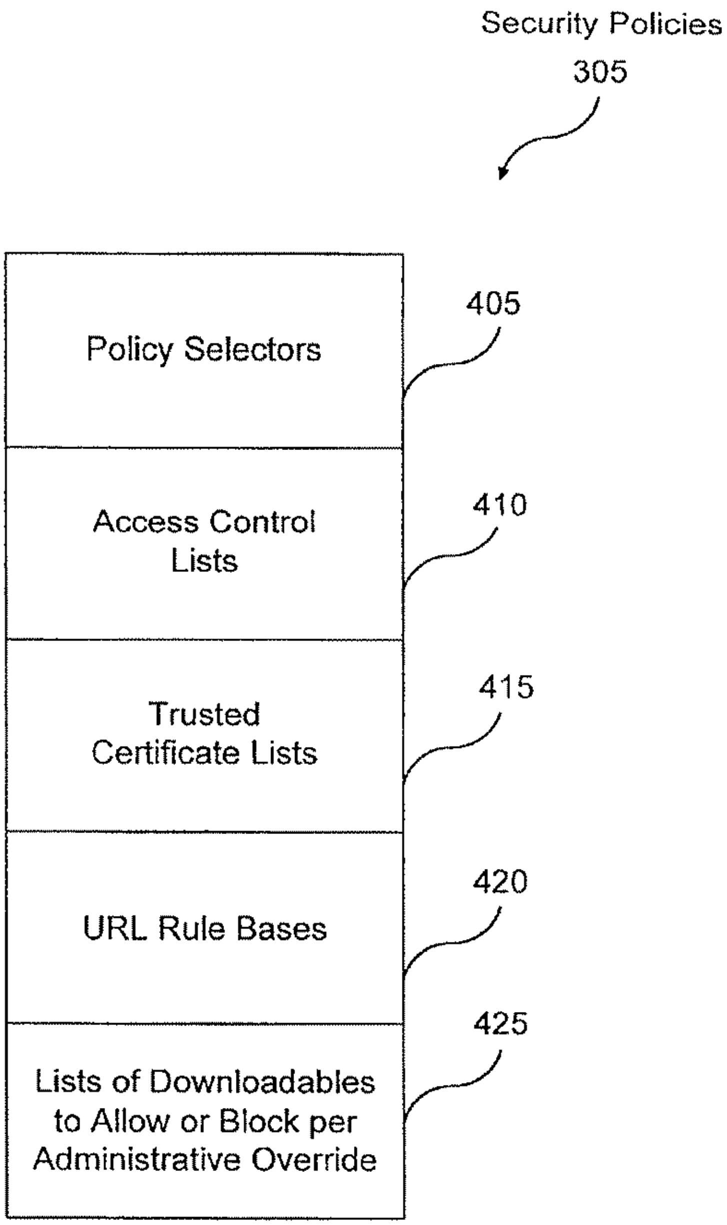

Security Policies
305
Policy Selectors
405
Access Control Lists
410
Trusted Certificate Lists
415
URL Rule Bases
420
Lists of Downloadables to Allow or Block per Administrative Override
425

FIG. 4

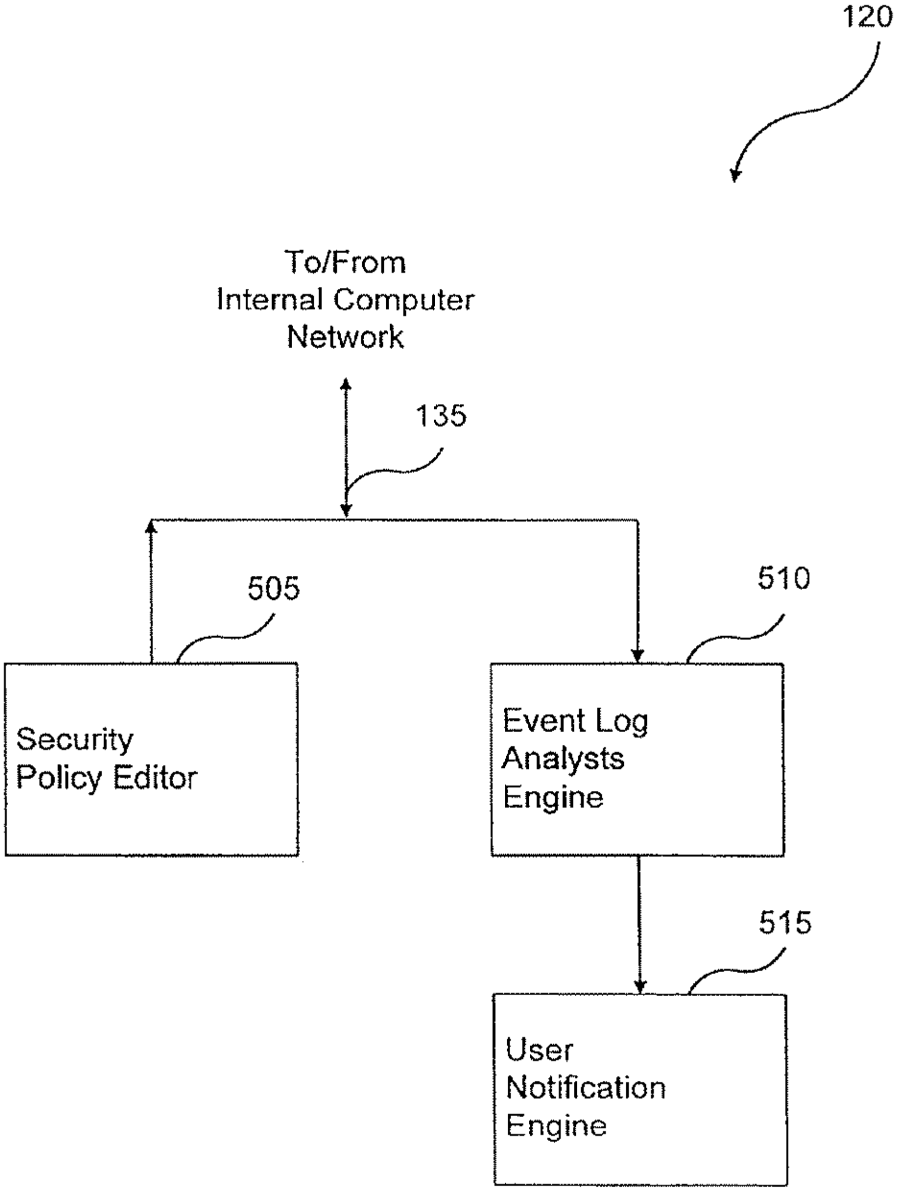

120
To/From
Internal Computer
Network
135
505
510
Security
Policy Editor
Event Log
Analysts
Engine
515
User
Notification
Engine

FIG. 5

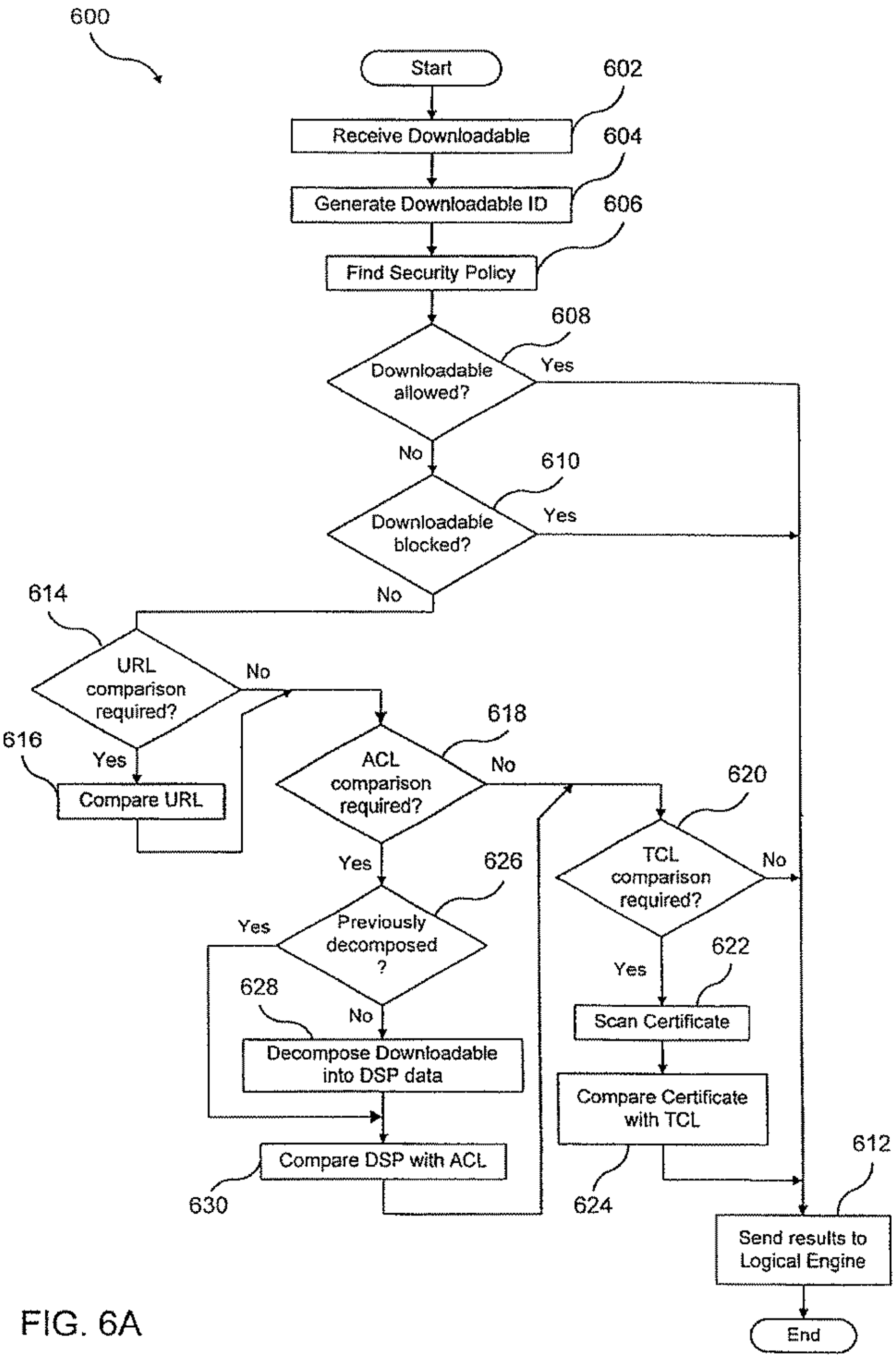

600
Start
602
Receive Downloadable
604
Generate Downloadable ID
606
Find Security Policy
608
Downloadable allowed?
Yes
No
610
Downloadable blocked?
Yes
No
614
URL comparison required?
No
616
Yes
Compare URL
618
ACL comparison required?
No
620
TCL comparison required?
No
Yes
626
Previously decomposed ?
Yes
628
No
Decompose Downloadable into DSP data
Compare DSP with ACL
630
Yes
622
Scan Certificate
Compare Certificate with TCL
624
612
Send results to Logical Engine
End

FIG. 6A

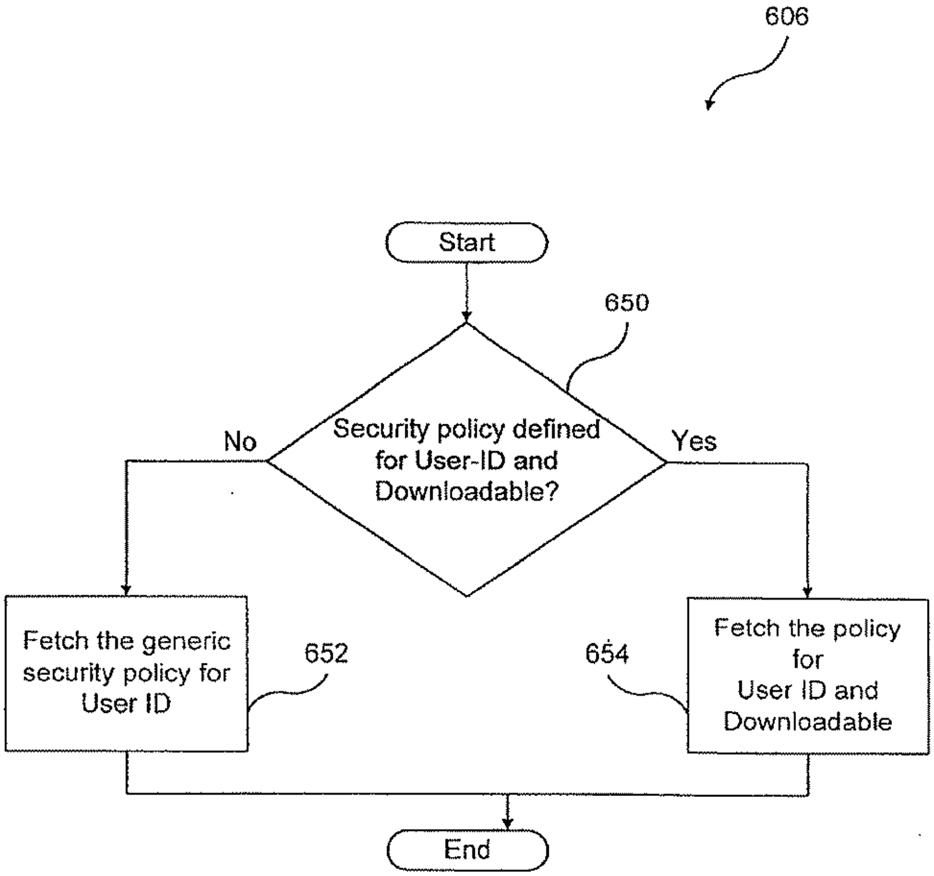

606
Start
650
Security policy defined for User-ID and Downloadable?
No
Yes
Fetch the generic security policy for User ID
652
654
Fetch the policy for User ID and Downloadable
End

FIG. 6B

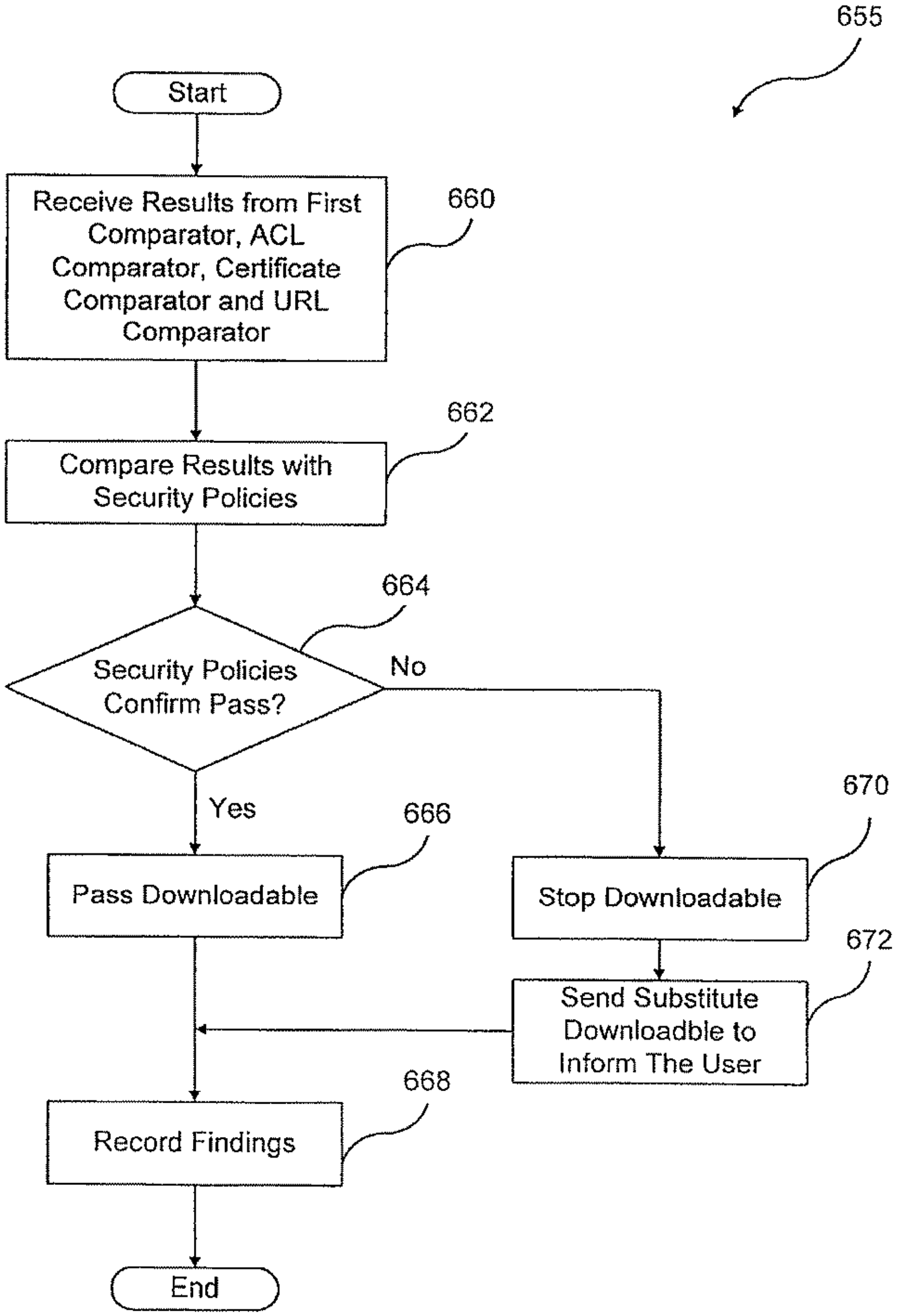

655
Start
Receive Results from First Comparator, ACL Comparator, Certificate Comparator and URL Comparator
660
Compare Results with Security Policies
662
664
Security Policies Confirm Pass?
No
Yes
666
Pass Downloadable
670
Stop Downloadable
672
Send Substitute Downloadble to Inform The User
668
Record Findings
End

FIG. 6C

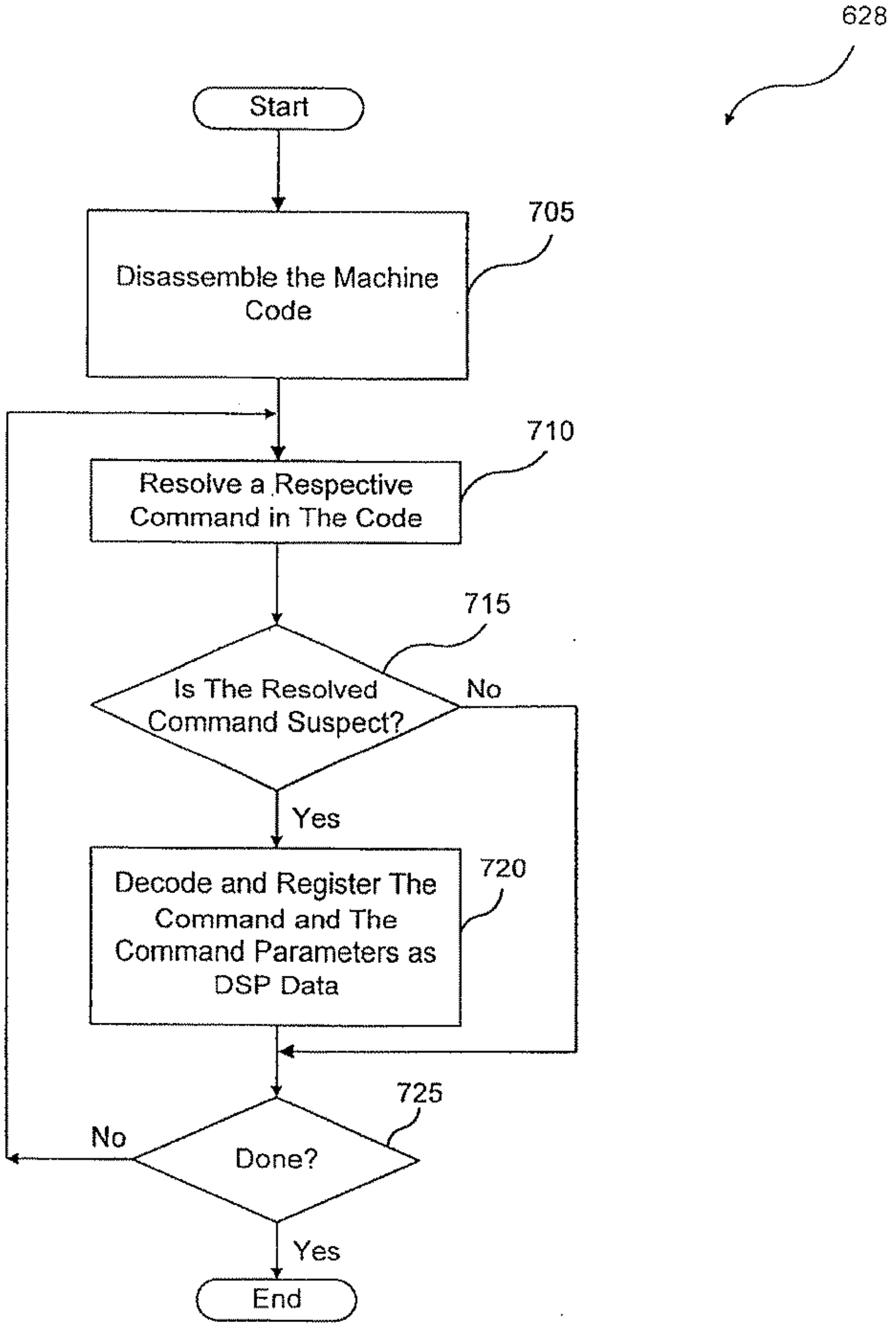

628
Start
Disassemble the Machine Code
705
Resolve a Respective Command in The Code
710
Is The Resolved Command Suspect?
715
No
Yes
Decode and Register The Command and The Command Parameters as DSP Data
720
Done?
725
No
Yes
End

FIG. 7

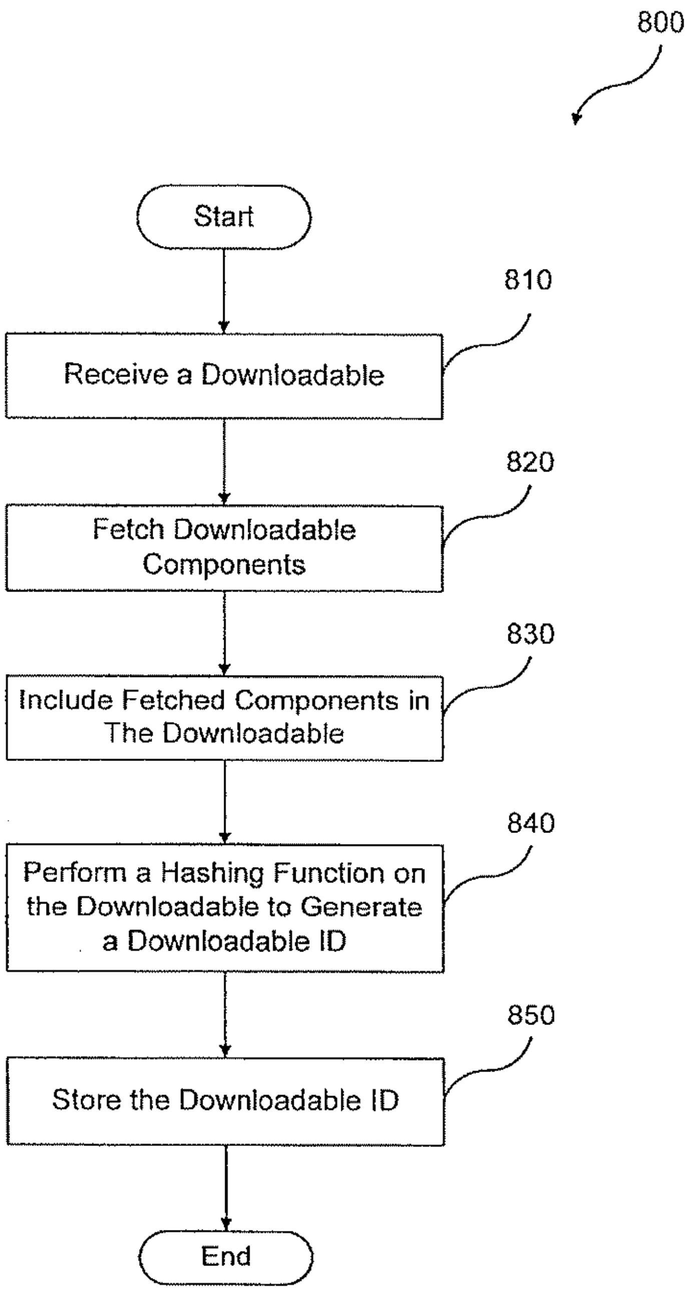

800
Start
810
Receive a Downloadable
820
Fetch Downloadable Components
830
Include Fetched Components in The Downloadable
840
Perform a Hashing Function on the Downloadable to Generate a Downloadable ID
850
Store the Downloadable ID
End

FIG. 8

SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES

PRIORITY REFERENCE TO RELATED APPLICATION

This application is a continuation of and hereby incorporates by reference U.S. patent application Ser. No. 08/964,388, entitled "System and Method for Protecting a Computer and a Network from Hostile Downloadables," filed Nov. 6, 1997, which is now U.S. Pat. No. 6,092,194, which claims priority to provisional application Serial No. 60/030,639, entitled "System and Method for Protecting a Computer from Hostile Downloadables," filed on Nov. 8, 1996, by inventor Shlomo Touboul.

INCORPORATION BY REFERENCE TO RELATED APPLICATIONS

This application hereby incorporates by reference related U.S. patent application Ser. No. 08/790,097, entitled "System and Method for Protecting a Client from Hostile Downloadables," filed on Jan. 29, 1997, which is now U.S. Pat. No. 6,167,520, by inventor Shlomo Touboul; and hereby incorporates by reference provisional application Ser. No. 60/030,639, entitled "System and Method for Protecting a Computer from Hostile Downloadables," filed on Nov. 8, 1996, by inventor Shlomo Touboul.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and method for protecting a computer and a network from hostile Downloadables.

2. Description of the Background Art

The Internet is currently a collection of over 100,000 individual computer networks owned by governments, universities, nonprofit groups and companies, and is expanding at an accelerating rate. Because the Internet is public, the Internet has become a major source of many system damaging and system fatal application programs, commonly referred to as "viruses."

Accordingly, programmers continue to design computer and computer network security systems for blocking these viruses from attacking both individual and network computers. On the most part, these security systems have been relatively successful. However, these security systems are not configured to recognize computer viruses which have been attached to or configured as Downloadable application programs, commonly referred to as "Downloadables." A Downloadable is an executable application program, which is downloaded from a source computer and run on the destination computer. Downloadable is typically requested by an ongoing process such as by an Internet browser or web engine. Examples of Downloadables include Java™ applets designed for use in the Java™ distributing environment developed by Sun Microsystems, Inc., JavaScript scripts also developed by Sun Microsystems, Inc., ActiveX™ controls designed for use in the ActiveX™ distributing environment developed by the Microsoft Corporation, and Visual Basic also developed by the Microsoft Corporation. Therefore, a system and method are needed to protect a network from hostile Downloadables.

SUMMARY OF THE INVENTION

The present invention provides a system for protecting a network from suspicious Downloadables. The system comprises a security policy, an interface for receiving a Downloadable, and a comparator, coupled to the interface, for applying the security policy to the Downloadable to determine if the security policy has been violated. The Downloadable may include a Java™ applet, an ActiveX™ control, a JavaScript™ script, or a Visual Basic script. The security policy may include a default security policy to be applied regardless of the client to whom the Downloadable is addressed, a specific security policy to be applied based on the client or the group to which the client belongs, or a specific policy to be applied based on the client/group and on the particular Downloadable received. The system uses an ID generator to compute a Downloadable ID identifying the Downloadable, preferably, by fetching all components of the Downloadable and performing a hashing function on the Downloadable including the fetched components.

Further, the security policy may indicate several tests to perform, including (1) a comparison with known hostile and non-hostile Downloadables; (2) a comparison with Downloadables to be blocked or allowed per administrative override; (3) a comparison of the Downloadable security profile data against access control lists; (4) a comparison of a certificate embodied in the Downloadable against trusted certificates; and (5) a comparison of the URL from which the Downloadable originated against trusted and untrusted URLs. Based on these tests, a logical engine can determine whether to allow or block the Downloadable.

The present invention further provides a method for protecting a computer from suspicious Downloadables. The method comprises the steps of receiving a Downloadable, comparing the Downloadable against a security policy to determine if the security policy has been violated, and discarding the Downloadable if the security policy has been violated.

It will be appreciated that the system and method of the present invention may provide computer protection from known hostile Downloadables. The system and method of the present invention may identify Downloadables that perform operations deemed suspicious. The system and method of the present invention may examine the Downloadable code to determine whether the code contains any suspicious operations, and thus may allow or block the Downloadable accordingly.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating a network system, in accordance with the present invention;

FIG. 2 is a block diagram illustrating details of the internal network security system of FIG. 1;

FIG. 3 is a block diagram illustrating details of the security program and the security database of FIG. 2;

FIG. 4 is a block diagram illustrating details of the security policies of FIG. 3;

FIG. 5 is a block diagram illustrating details of the security management console of FIG. 1;

FIG. 6A is a flowchart illustrating a method of examining for suspicious Downloadables, in accordance with the present invention;

FIG. 6B is a flowchart illustrating details of the step for finding the appropriate security policy of FIG. 6A;

FIG. 6C is a flowchart illustrating a method for determining whether an incoming Downloadable is to be deemed suspicious;

FIG. 7 is a flowchart illustrating details of the FIG. 6 step of decomposing a Downloadable; and

FIG. 8 is a flowchart illustrating a method 800 for generating a Downloadable ID for identifying a Downloadable.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 is a block diagram illustrating a network system 100, in accordance with the present invention. The network system 100 includes an external computer network 105, such as the Wide Area Network (WAN) commonly referred to as the Internet, coupled via a communications channel 125 to an internal network security system 110. The network system 100 further includes an internal computer network 115, such as a corporate Local Area Network (LAN), coupled via a communications channel 130 to the internal network computer system 110 and coupled via a communications channel 135 to a security management console 120.

The internal network security system 110 examines Downloadables received from external computer network 105, and prevents Downloadables deemed suspicious from reaching the internal computer network 115. It will be further appreciated that a Downloadable is deemed suspicious if it performs or may perform any undesirable operation, or if it threatens or may threaten the integrity of an internal computer network 115 component. It is to be understood that the term "suspicious" includes hostile, potentially hostile, undesirable, potentially undesirable, etc. Security management console 120 enables viewing, modification and configuration of the internal network security system 110.

FIG. 2 is a block diagram illustrating details of the internal network security system 110, which includes a Central Processing Unit (CPU) 205, such as an Intel Pentium® microprocessor or a Motorola Power PC® microprocessor, coupled to a signal bus 220. The internal network security system 110 further includes an external communications interface 210 coupled between the communications channel 125 and the signal bus 220 for receiving Downloadables from external computer network 105, and an internal communications interface 225 coupled between the signal bus 220 and the communications channel 130 for forwarding Downloadables not deemed suspicious to the internal computer network 115. The external communications interface 210 and the internal communications interface 225 may be functional components of an integral communications interface (not shown) for both receiving Downloadables from the external computer network 105 and forwarding Downloadables to the internal computer network 115.

Internal network security system 110 further includes Input/Output (I/O) interfaces 215 (such as a keyboard, mouse and Cathode Ray Tube (CRT) display), a data storage device 230 such as a magnetic disk, and a Random-Access Memory (RAM) 235, each coupled to the signal bus 220. The data storage device 230 stores a security database 240, which includes security information for determining whether a received Downloadable is to be deemed suspicious. The data storage device 230 further stores a users list 260 identifying the users within the internal computer network 115 who may receive Downloadables, and an event log 245 which includes determination results for each Downloadable examined and runtime indications of the internal network security system 110. An operating system 250 controls processing by CPU 205, and is typically stored in data storage device 230 and loaded into RAM 235 (as illustrated) for execution. A security program 255 controls

examination of incoming Downloadables, and also may be stored in data storage device 230 and loaded into RAM 235 (as illustrated) for execution by CPU 205.

FIG. 3 is a block diagram illustrating details of the security program 255 and the security database 240. The security program 255 includes an ID generator 315, a policy finder 317 coupled to the ID generator 315, and a first comparator 320 coupled to the policy finder 317. The first comparator 320 is coupled to a logical engine 333 via four separate paths, namely, via Path 1, via Path 2, via Path 3 and via Path 4. Path 1 includes a direct connection from the first comparator 320 to the logical engine 333. Path 2 includes a code scanner coupled to the first comparator 320, and an Access Control List (ACL) comparator 330 coupling the code scanner 325 to the logical engine 333. Path 3 includes a certificate scanner 340 coupled to the first comparator 320, and a certificate comparator 345 coupling the certificate scanner 340 to the logical engine 333. Path 4 includes a Uniform Resource Locator (URL) comparator 350 coupling the first comparator 320 to the logical engine 3330. A record-keeping engine 335 is coupled between the logical engine 333 and the event log 245.

The security program 255 operates in conjunction with the security database 240, which includes security policies 305, known Downloadables 307, known Certificates 309 and Downloadable Security Profile (DSP) data 310 corresponding to the known Downloadables 307. Security policies 305 includes policies specific to particular users 260 and default (or generic) policies for determining whether to allow or block an incoming Downloadable. These security policies 305 may identify specific Downloadables to block, specific Downloadables to allow, or necessary criteria for allowing an unknown Downloadable. Referring to FIG. 4, security policies 305 include policy selectors 405, access control lists 410, trusted certificate lists 415, URL rule bases 420, and lists 425 of Downloadables to allow or to block per administrative override.

Known Downloadables 307 include lists of Downloadables which Original Equipment Manufacturers (OEMs) know to be hostile, of Downloadables which OEMs know to be non-hostile, and of Downloadables previously received by this security program 255. DSP data 310 includes the list of all potentially hostile or suspicious computer operations that may be attempted by each known Downloadable 307, and may also include the respective arguments of these operations. An identified argument of an operation is referred to as "resolved." An unidentified argument is referred to as "unresolved." DSP data 310 is described below with reference to the code scanner 325.

The ID generator 315 receives a Downloadable (including the URL from which it came and the userID of the intended recipient) from the external computer network 105 via the external communications interface 210, and generates a Downloadable ID for identifying each Downloadable. The Downloadable ID preferably includes a digital hash of the complete Downloadable code. The ID generator 315 preferably prefetches all components embodied in or identified by the code for Downloadable ID generation. For example, the ID generator 315 may prefetch all classes embodied in or identified by the Java™ applet bytecode to generate the Downloadable ID. Similarly, the ID generator 315 may retrieve all components listed in the INF file for an ActiveX™ control to compute a Downloadable ID. Accordingly, the Downloadable ID for the Downloadable will be the same each time the ID generator 315 receives the same Downloadable. The ID generator 315 adds the generated Downloadable ID to the list of known Downloadables

307 (if it is not already listed). The ID generator 315 then forwards the Downloadable and Downloadable ID to the policy finder 317.

The policy finder 317 uses the userID of the intended user and the Downloadable ID to select the specific security policy 305 that shall be applied on the received Downloadable. If there is a specific policy 305 that was defined for the user (or for one of its super groups) and the Downloadable, then the policy is selected. Otherwise the generic policy 305 that was defined for the user (or for one of its super groups) is selected. The policy finder 317 then sends the policy to the first comparator 320.

The first comparator 320 receives the Downloadable, the Downloadable ID and the security policy 305 from the policy finder 317. The first comparator 320 examines the security policy 305 to determine which steps are needed for allowing the Downloadable. For example, the security policy 305 may indicate that, in order to allow this Downloadable, it must pass all four paths, Path 1, Path 2, Path 3 and Path 4. Alternatively, the security policy 305 may indicate that to allow the Downloadable, the it must pass only one of the paths. The first comparator 320 responds by forwarding the proper information to the paths identified by the security policy 305.

Path 1

In path 1, the first comparator 320 checks the policy selector 405 of the security policy 305 that was received from the policy finder 317. If the policy selector 405 is either "Allowed" or "Blocked," then the first comparator 320 forwards this result directly to the logical engine 333. Otherwise, the first comparator 320 invokes the comparisons in path2 and/or path 3 and/or path 4 based on the contents of policy selector 405. It will be appreciated that the first comparator 320 itself compares the Downloadable ID against the lists of Downloadables to allow or block per administrative override 425. That is, the system security administrator can define specific Downloadables as "Allowed" or "Blocked."

Alternatively, the logical engine 333 may receive the results of each of the paths and based on the policy selector 405 may institute the final determination whether to allow or block the Downloadable. The first comparator 320 informs the logical engine 333 of the results of its comparison.

Path 2

In path 2, the first comparator 320 delivers the Downloadable, the Downloadable ID and the security policy 305 to the code scanner 325. If the DSP data 310 of the received Downloadable is known, the code scanner 325 retrieves and forwards the information to the ACL comparator 330. Otherwise, the code scanner 325 resolves the DSP data 310. That is, the code scanner 325 uses conventional parsing techniques to decompose the code (including all prefetched components) of the Downloadable into the DSP data 310. DSP data 310 includes the list of all potentially hostile or suspicious computer operations that may be attempted by a specific Downloadable 307, and may also include the respective arguments of these operations. For example, DSP data 310 may include a READ from a specific file, a SEND to an unresolved host, etc. The code scanner 325 may generate the DSP data 310 as a list of all operations in the Downloadable code which could ever be deemed potentially hostile and a list of all files to be accessed by the Downloadable code. It will be appreciated that the code scanner 325 may search the code for any pattern, which is undesirable or suggests that the code was written by a hacker.

An Example List of Operations Deemed Potentially Hostile

File operations: READ a file, WRITE a file;

Network operations: LISTEN on a socket, CONNECT to a socket, SEND data, RECEIVE data, VIEW INTRANET;

Registry operations: READ a registry item, WRITE a registry item;

Operating system operations: EXIT WINDOWS, EXIT BROWSER, START PROCESS/THREAD, KILL PROCESS/THREAD, CHANGE PROCESS/THREAD PRIORITY, DYNAMICALLY LOAD A CLASS/LIBRARY, etc.; and

Resource usage thresholds: memory, CPU, graphics, etc.

In the preferred embodiment, the code scanner 325 performs a full-content inspection. However, for improved speed but reduced security, the code scanner 325 may examine only a portion of the Downloadable such as the Downloadable header. The code scanner 325 then stores the DSP data into DSP data 310 (corresponding to its Downloadable ID), and sends the Downloadable, the DSP data to the ACL comparator 330 for comparison with the security policy 305.

The ACL comparator 330 receives the Downloadable, the corresponding DSP data and the security policy 305 from the code scanner 325, and compares the DSP data against the security policy 305. That is, the ACL comparator 330 compares the DSP data of the received Downloadable against the access control lists 410 in the received security policy 305. The access control list 410 contains criteria indicating whether to pass or fail the Downloadable. For example, an access control list may indicate that the Downloadable fails if the DSP data includes a WRITE command to a system file. The ACL comparator 330 sends its results to the logical engine 333.

Path 3

In path 3, the certificate scanner 340 determines whether the received Downloadable was signed by a certificate authority, such as VeriSign, Inc., and scans for a certificate embodied in the Downloadable. The certificate scanner 340 forwards the found certificate to the certificate comparator 345. The certificate comparator 345 retrieves known certificates 309 that were deemed trustworthy by the security administrator and compares the found certificate with the known certificates 309 to determine whether the Downloadable was signed by a trusted certificate. The certificate comparator 345 sends the results to the logical engine 333.

Path 4

In path 4, the URL comparator 350 examines the URL identifying the source of the Downloadable against URLs stored in the URL rule base 420 to determine whether the Downloadable comes from a trusted source. Based on the security policy 305, the URL comparator 350 may deem the Downloadable suspicious if the Downloadable comes from an untrustworthy source or if the Downloadable did not come from a trusted source. For example, if the Downloadable comes from a known hacker, then the Downloadable may be deemed suspicious and presumed hostile. The URL comparator 350 sends its results to the logical engine 333.

The logical engine 333 examines the results of each of the paths and the policy selector 405 in the security policy 305 to determine whether to allow or block the Downloadable. The policy selector 405 includes a logical expression of the results received from each of the paths. For example, the logical engine 333 may block a Downloadable if it fails any one of the paths, i.e., if the Downloadable is known hostile (Path 1), if the Downloadable may request suspicious operations (Path 2), if the Downloadable was not signed by a trusted certificate authority (Path 3), or if the Downloadable did came from an untrustworthy source (Path 4). The logical engine 333 may apply other logical expressions according to the policy selector 405 embodied in the security policy 305. If the policy selector 405 indicates that the Downloadable may pass, then the logical engine 333 passes the Downloadable to its intended recipient. Otherwise, if the policy selector 405 indicates that the Downloadable should be blocked, then the logical engine 333 forwards a non-hostile Downloadable to the intended recipient to inform the user that internal network security system 110 discarded the original Downloadable. Further, the logical engine 333 forwards a status report to the record-keeping engine 335, which stores the reports in event log 245 in the data storage device 230 for subsequent review, for example, by the MIS director.

FIG. 5 is a block diagram illustrating details of the security management console 120, which includes a security policy editor 505 coupled to the communications channel 135, an event log analysis engine 510 coupled between communications channel 135 and a user notification engine 515, and a Downloadable database review engine 520 coupled to the communications channel 135. The security management console 120 further includes computer components similar to the computer components illustrated in FIG. 2.

The security policy editor 505 uses an I/O interface similar to I/O interface 215 for enabling authorized user modification of the security policies 305. That is, the security policy editor 505 enables the authorized user to modify specific security policies 305 corresponding to the users 260, the default or generic security policy 305, the Downloadables to block per administrative override, the Downloadables to allow per administrative override, the trusted certificate lists 415, the policy selectors 405, the access control lists 410, the URLs in the URL rule bases 420, etc. For example, if the authorized user learns of a new hostile Downloadable, then the user can add the Downloadable to the Downloadables to block per system override.

The event log analysis engine 510 examines the status reports contained in the event log 245 stored in the data storage device 230. The event log analysis engine 510 determines whether notification of the user (e.g., the security system manager or MIS director) is warranted. For example, the event log analysis engine 510 may warrant user notification whenever ten (10) suspicious Downloadables have been discarded by internal network security system 110 within a thirty (30) minute period, thereby flagging a potential imminent security threat. Accordingly, the event log analysis engine 510 instructs the user notification engine 515 to inform the user. The user notification engine 515 may send an e-mail via internal communications interface 220 or via external communications interface 210 to the user, or may display a message on the user's display device (not shown).

FIG. 6A is a flowchart illustrating a method 600 for protecting an internal computer network 115 from suspicious Downloadables. Method 600 begins with the ID generator 315 in step 602 receiving a Downloadable. The ID generator 315 in step 604 generates a Downloadable ID identifying the received Downloadable, preferably, by generating a digital hash of the Downloadable code (including prefetched components). The policy finder 317 in step 606 finds the appropriate security policy 305 corresponding to the userID specifying intended recipient (or the group to which the intended recipient belongs) and the Downloadable. The selected security policy 305 may be the default security policy 305. Step 606 is described in greater detail below with reference to FIG. 6B.

The first comparator 320 in step 608 examines the lists of Downloadables to allow or to block per administrative override 425 against the Downloadable ID of the incoming Downloadable to determine whether to allow the Downloadable automatically. If so, then in step 612 the first comparator 320 sends the results to the logical engine 333. If not, then the method 600 proceeds to step 610. In step 610, the first comparator 620 examines the lists of Downloadables to block per administrative override 425 against the Downloadable ID of the incoming Downloadable for determining whether to block the Downloadable automatically. If so, then the first comparator 420 in step 612 sends the results to the logical engine 333. Otherwise, method 600 proceeds to step 614.

In step 614, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 4. If not, then method 600 jumps to step 618. If so, then the URL comparator 350 in step 616 compares the URL embodied in the incoming Downloadable against the URLs of the URL rules bases 420, and then method 600 proceeds to step 618.

In step 618, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 2. If not, then method 600 jumps to step 620. Otherwise, the code scanner 235 in step 626 examines the DSP data 310 based on the Downloadable ID of the incoming Downloadable to determine whether the Downloadable has been previously decomposed. If so, then method 600 jumps to step 630. Otherwise, the code scanner 325 in step 628 decomposes the Downloadable into DSP data. Downloadable decomposition is described in greater detail with reference to FIG. 7. In step 630, the ACL comparator 330 compares the DSP data of the incoming Downloadable against the access control lists 410 (which include the criteria necessary for the Downloadable to fail or pass the test).

In step 620, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 3. If not, then method 600 returns to step 612 to send the results of each of the test performed to the logical engine 333. Otherwise, the certificate scanner 622 in step 622 scans the Downloadable for an embodied certificate. The certificate comparator 345 in step 624 retrieves trusted certificates from the trusted certificate lists (TCL) 415 and compares the embodied certificate with the trusted certificates to determine whether the Downloadable has been signed by a trusted source. Method 600 then proceeds to step 612 by the certificate scanner 345 sending the results of each of the paths taken to the logical engine 333. The operations of the logical engine 333 are described in greater detail below with reference to FIG. 6C. Method 600 then ends.

One skilled in the art will recognize that the tests may be performed in a different order, and that each of the tests need not be performed. Further, one skilled in the art will recognize that, although path 1 is described in FIG. 6A as an automatic allowance or blocking, the results of Path 1 may be another predicate to be applied by the logical engine 333. Further, although the tests are shown serially in FIG. 6A, the tests may be performed in parallel as illustrated in FIG. 3.

FIG. **6B** is a flowchart illustrating details of step **606** of FIG. **6A** (referred to herein as method **606**). Method **606** begins with the policy finder **317** in step **650** determining whether security policies **305** include a specific security policy corresponding to the userID and the Downloadable. If so, then the policy finder **317** in step **654** fetches the corresponding specific policy **305**. If not, then the policy finder **317** in step **652** fetches the default or generic security policy **305** corresponding to the userID. Method **606** then ends.

FIG. **6C** is a flowchart illustrating details of a method **655** for determining whether to allow or to block the incoming Downloadable. Method **655** begins with the logical engine **333** in step **660** receiving the results from the first comparator **320**, from the ACL comparator **330**, from the certificate comparator **345** and from the URL comparator **350**. The logical engine **333** in step **662** compares the results with the policy selector **405** embodied in the security policy **305**, and in step **664** determines whether the policy selector **405** confirms the pass. For example, the policy selector **405** may indicate that the logical engine **333** pass the Downloadable if it passes one of the tests of Path **1**, Path **2**, Path **3** and Path **4**. If the policy selector **405** indicates that the Downloadable should pass, then the logical engine **333** in step **666** passes the Downloadable to the intended recipient. In step **668**, the logical engine **333** sends the results to the record-keeping engine **335**, which in turn stores the results in the event log **245** for future review. Method **655** then ends. Otherwise, if the policy selector **405** in step **664** indicates that the Downloadable should not pass, then the logical engine **333** in step **670** stops the Downloadable and in step **672** sends a non-hostile substitute Downloadable to inform the user that the incoming Downloadable has been blocked. Method **655** then jumps to step **668**.

FIG. **7** is a flowchart illustrating details of step **628** of FIG. **6A** (referred to herein as method **628**) for decomposing a Downloadable into DSP data **310**. Method **628** begins in step **705** with the code scanner **325** disassembling the machine code of the Downloadable. The code scanner **325** in step **710** resolves a respective command in the machine code, and in step **715** determines whether the resolved command is suspicious (e.g., whether the command is one of the operations identified in the list described above with reference to FIG. **3**). If not, then the code scanner **325** in step **725** determines whether it has completed decomposition of the Downloadable, i.e., whether all operations in the Downloadable code have been resolved. If so, then method **628** ends. Otherwise, method **628** returns to step **710**.

Otherwise, if the code scanner **325** in step **71** determines that the resolved command is suspect, then the code scanner **325** in step **720** decodes and registers the suspicious command and its command parameters as DSP data **310**. The code scanner **325** in step **720** registers the commands and command parameters into a format based on command class (e.g., file operations, network operations, registry operations, operating system operations, resource usage thresholds). Method **628** then jumps to step **725**.

FIG. **8** is a flowchart illustrating a method **800** for generating a Downloadable ID for identifying a Downloadable. Method **800** begins with the ID generator **315** in step **810** receiving a Downloadable from the external computer network **105**. The ID generator **315** in step **820** may fetch some or all components referenced in the Downloadable code, and in step **830** includes the fetched components in the Downloadable code. The ID generator **315** in step **840** performs a hashing function on at least a portion of the Downloadable code to generate a Downloadable ID. The ID generator **315** in step **850** stores the generated Downloadable ID in the security database **240** as a reference to the DSP data **310**. Accordingly, the Downloadable ID will be the same for the identical Downloadable each time it is encountered.

The foregoing description of the preferred embodiments of the invention is by way of example only, and other variations of the above-described embodiments and methods are provided by the present invention. For example, although the invention has been described in a system for protecting an internal computer network, the invention can be embodied in a system for protecting an individual computer. Components of this invention may be implemented using a programmed general purpose digital computer, using application specific integrated circuits, or using a network of interconnected conventional components and circuits. The embodiments described herein have been presented for purposes of illustration and are not intended to be exhaustive or limiting. Many variations and modifications are possible in light of the foregoing teaching. The system is limited only by the following claims.

What is claimed is:

1. A computer-based method for generating a Downloadable ID to identify a Downloadable, comprising:

obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;

fetching at least one software component identified by the one or more references; and

performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

2. The method of claim **1**, wherein the Downloadable includes an applet.

3. The method of claim **1**, wherein the Downloadable includes an active software control.

4. The method of claim **1**, wherein the Downloadable includes a plugin.

5. The method of claim **1**, wherein the Downloadable includes HTML code.

6. The method of claim **1**, wherein the Downloadable includes an application program.

7. The method of claim **1**, wherein said fetching includes fetching a first software component referenced by the Downloadable.

8. The method of claim **1**, wherein said fetching includes fetching all software components referenced by the Downloadable.

9. A system for generating a Downloadable ID to identify a Downloadable, comprising:

a communications engine for obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable; and

an ID generator coupled to the communications engine that fetches at least one software component identified by the one or more references, and for performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

10. The system of claim **9**, wherein the Downloadable includes an applet.

11. The system of claim **9**, wherein the Downloadable includes an active software control.

12. The system of claim **9**, wherein the Downloadable includes a plugin.

13. The system of claim **9**, wherein the Downloadable includes HTML code.

14. The system of claim 9, wherein the Downloadable includes an application program.

15. The system of claim 9, wherein the ID generator fetches a first software component referenced by the Downloadable.

16. The method of claim 9, wherein the ID generator fetches all software components referenced by the Downloadable.

17. A system for generating a Downloadable ID to identify a Downloadable, comprising:

means for obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;

means for fetching at least one software component identified by the one or more references; and

means for performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

18. A computer-readable storage medium storing program code for causing a computer to perform the steps of:

obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;

fetching at least one software component identified by the one or more references; and

performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

* * * * *

EXHIBIT C

US007058822B2

(12) United States Patent
Edery et al.

(10) **Patent No.:** **US 7,058,822 B2**
(45) **Date of Patent:** **Jun. 6, 2006**

(54) **MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS**

(75) Inventors: **Yigal Mordechai Edery**, Pardesia (IL); **Nimrod Itzhak Vered**, Goosh Tel-Mond (IL); **David R. Kroll**, San Jose, CA (US)

(73) Assignee: **Finjan Software, Ltd.**, South Netanya (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1013 days.

(21) Appl. No.: **09/861,229**

(22) Filed: **May 17, 2001**

(65) **Prior Publication Data**

US 2002/0013910 A1 Jan. 31, 2002

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/551,302, filed on Apr. 18, 2000, now Pat. No. 6,480,962, which is a continuation-in-part of application No. 09/539, 667, filed on Mar. 30, 2000, now Pat. No. 6,804,780.

(60) Provisional application No. 60/205,591, filed on May 17, 2000.

(51) **Int. Cl.**
G06F 11/30 (2006.01)

(52) **U.S. Cl.** **713/200**

(58) **Field of Classification Search** 713/176, 713/175, 200, 201, 150, 168; 701/223, 229; 717/120, 124, 126, 127, 130, 131, 134, 135; 709/223–229
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,077,677	A	12/1991	Murphy et al.
5,359,659	A	10/1994	Rosenthal
5,361,359	A	11/1994	Tajalli et al.
5,485,409	A	1/1996	Gupta et al.

(Continued)

OTHER PUBLICATIONS

Zhong et al, "Security in the large: is Java's sandbox scalable?", Oct. 1998, Seventh IEEE Symposium on Reliable Distributed Systems, pp 1-6.*

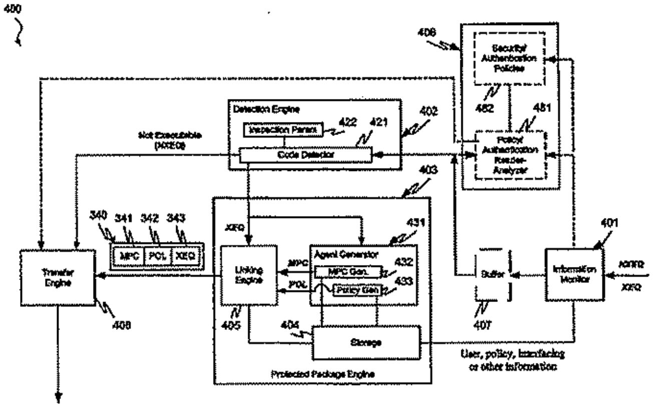
(Continued)

Primary Examiner—Christopher Revak
(74) *Attorney, Agent, or Firm*—Squire, Sanders & Dempsey, L.L.P.

(57) **ABSTRACT**

Protection systems and methods provide for protecting one or more personal computers ("PCs") and/or other intermittently or persistently network accessible devices or processes from undesirable or otherwise malicious operations of Java™ applets, ActiveX™ controls, JavaScript™ scripts, Visual Basic scripts, add-ins, downloaded/uploaded programs or other "Downloadables" or "mobile code" in whole or part. A protection engine embodiment provides, within a server, firewall or other suitable "re-communicator," for monitoring information received by the communicator, determining whether received information does or is likely to include executable code, and if so, causes mobile protection code (MPC) to be transferred to and rendered operable within a destination device of the received information, more suitably by forming a protection agent including the MPC, protection policies and a detected-Downloadable. An MPC embodiment further provides, within a Downloadable-destination, for initiating the Downloadable, enabling malicious Downloadable operation attempts to be received by the MPC, and causing (predetermined) corresponding operations to be executed in response to the attempts, more suitably in conjunction with protection policies.

35 Claims, 10 Drawing Sheets



U.S. PATENT DOCUMENTS

5,485,575	A	1/1996	Chess et al.	
5,572,643	A	11/1996	Judson	
5,606,668	A	2/1997	Shwed	
5,623,600	A	4/1997	Ji et al.	
5,638,446	A	6/1997	Rubin	
5,692,047	A	11/1997	McManis	
5,692,124	A	11/1997	Holden et al.	
5,720,033	A	2/1998	Deo	
5,724,425	A	3/1998	Chang et al.	
5,740,248	A	4/1998	Fieres et al.	
5,761,421	A	6/1998	van Hoff et al.	
5,765,205	A	6/1998	Breslau et al.	
5,784,459	A	7/1998	Devarakonda et al.	
5,796,952	A	8/1998	Davis et al.	
5,805,829	A	9/1998	Cohen et al.	
5,832,208	A	11/1998	Chen et al.	
5,850,559	A	12/1998	Angelo et al.	
5,859,966	A	1/1999	Hayman et al.	
5,864,683	A	1/1999	Boebert et al.	
5,892,904	A	4/1999	Atkinson et al.	
5,951,698	A	9/1999	Chen et al.	
5,956,481	A	9/1999	Walsh et al.	
5,974,549	A	10/1999	Golan	
5,978,484	A	11/1999	Apperson et al.	
5,983,348	A	11/1999	Ji	
6,092,194	A	7/2000	Touboul	
6,154,844	A	11/2000	Touboul et al.	
6,167,520	A	12/2000	Touboul	
6,425,058	B1	7/2002	Arimilli et al.	
6,434,668	B1	8/2002	Arimilli et al.	
6,434,669	B1	8/2002	Arimilli et al.	
6,480,962	B1	11/2002	Touboul	
6,519,679	B1	2/2003	Devireddy et al.	
6,732,179	B1 *	5/2004	Brown et al.	709/229

OTHER PUBLICATIONS

Rubin et al, "Mobile code security" Dec. 1998, IEEE Internet, pp 30-34.*

Schmid et al, "Protecting data from malicious software", 2002, Proceeding of the 18th Annual Computer Security Applications Conference, pp 1-10.*

Corradi et al, "A flexible access control service for Java mobile code", 2000, IEEE, pp 356-365.*

Jim K. Omura, "Novel Applications of Cryptography in Digital Communications", IEEE Communications Magazine, May, 1990; pp. 21-29.

Okamoto, E. et al., "ID-Based Authentication System For Computer Virus Detection", IEEE/IEE Electronic Library online, Electronics Letters, vol. 26, Issue 15, ISSN 0013-5194, Jul. 19, 1990, Abstract and pp. 1169-1170. URL:http://iel.ihs.com:80/cgi-bin/iel_cgl?se...
2ehts%26ViewTemplate%3ddocview%5fb%2ehts.

IBM AntiVirus User's Guide Version 2.4, International Business Machines Corporation, Nov. 15, 1995, p. 6-7.

Norvin Leach et al, "IE 3.0 Applets Will Earn Certification", PC Week vol. 13, No. 29, Jul. 22, 1996, 2 pages.

"Finjan Software Releases SurfinBoard, Industry's First JAVA Security Product For the World Wide Web", Article published on the Internet by Finjan Software Ltd., Jul. 29, 1996, 1 page.

"Powerful PC Security for the New World of Java™ and Downloadables, Surfin Shield™" Article published on the Internet by Finjan Software Ltd., 1996, 2 Pages.

Microsoft® Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet", Microsoft Corporation, Oct. 1996, including Abstract, Contents, Introduction and pp. 1-10.

"Finjan Announces a Personal Java™ Firewall For Web Browsers—the SunfinShield™ 1.6 (formerly known as SurfinBoard)", Press Release of Finjan Releases SurfinShield 1.6, Oct. 21, 1996, 2 pages.

Company Profile "Finjan—Safe Surfing, The Java Security Solutions Provider" Article published on the Internet by Finjan Software Ltd., Oct. 31, 1996, 3 pages.

"Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0" Las Vegas Convention Center/Pavilion 5 P5551, Nov. 18, 1996, 3 pages.

"Java Security: Issues & Solutions" Articles published on the Internet by Finjan Software Ltd., 1996, 8 pages.

"Products" Articles published on the Internet, 7 pages.

Mark LaDue, "Online Business Consultant: Java Security: Whose Business Is It?" Article published on the Internet, Home Page Press, Inc. 1996, 4 pages.

Ron Moritz, "Why We Shouldn't Fear Java." Java Report, Feb., 1997, pp. 51-56.

Web Page Article "Frequently Asked Questions About Authenticode", Microsoft Corporation, last updated Feb. 17, 1997, Printed Dec. 23, 1998. URL: http://www.microsoft.com/workshop/security/authcode/signfag.asp#9, pp. 1-13.

Zhang, X.N., "Secure Code Distrubtion", IEEE/IEE Electronic Library online, Computer, vol. 30, Issue 6, Jun., 1997, pp.: 76-79.

Khare, Rohit, "Microsoft Authenticode Analyzed", Jul. 22, 1996, 2 pages. URL: http://www.xent.com/FoRK-archive/summer96/0338.html.

"Release Notes for the Microsoft ActiveX Development Kit", Aug. 13, 1996, 11 pages URL: http://activex.adsp.or.jp/inetsdk/readme.txt.

"Microsoft ActiveX Software Development Kit", Aug. 12, 1996, 6 pages. URL: http://activex.adsp.or.jp/inetsdk/help.overview.htm.

* cited by examiner

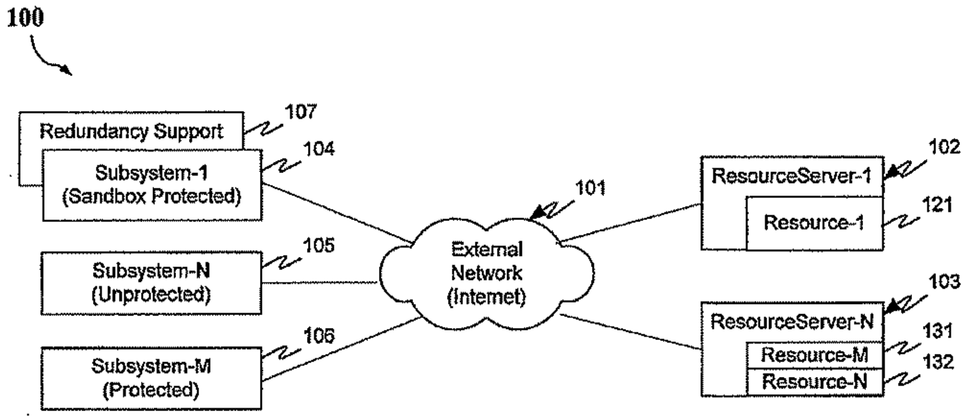
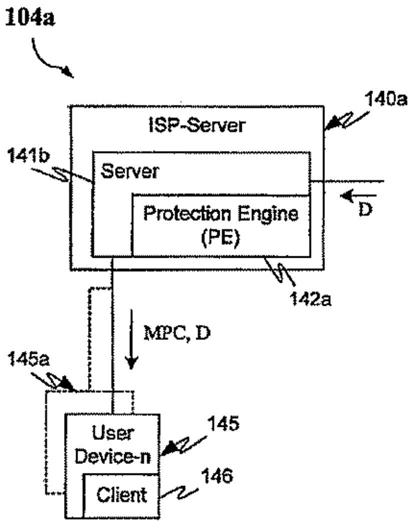


FIG. 1a



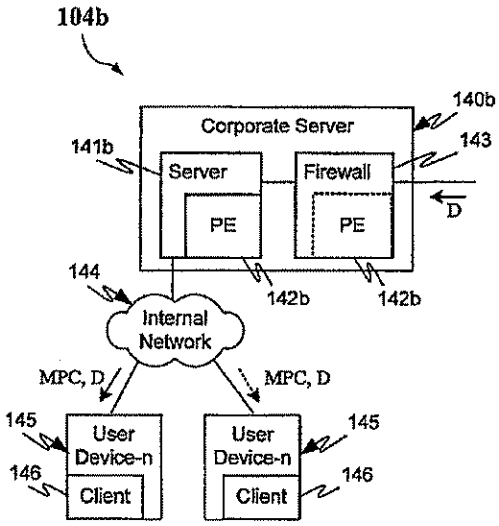


FIG. 1b

FIG. 1c

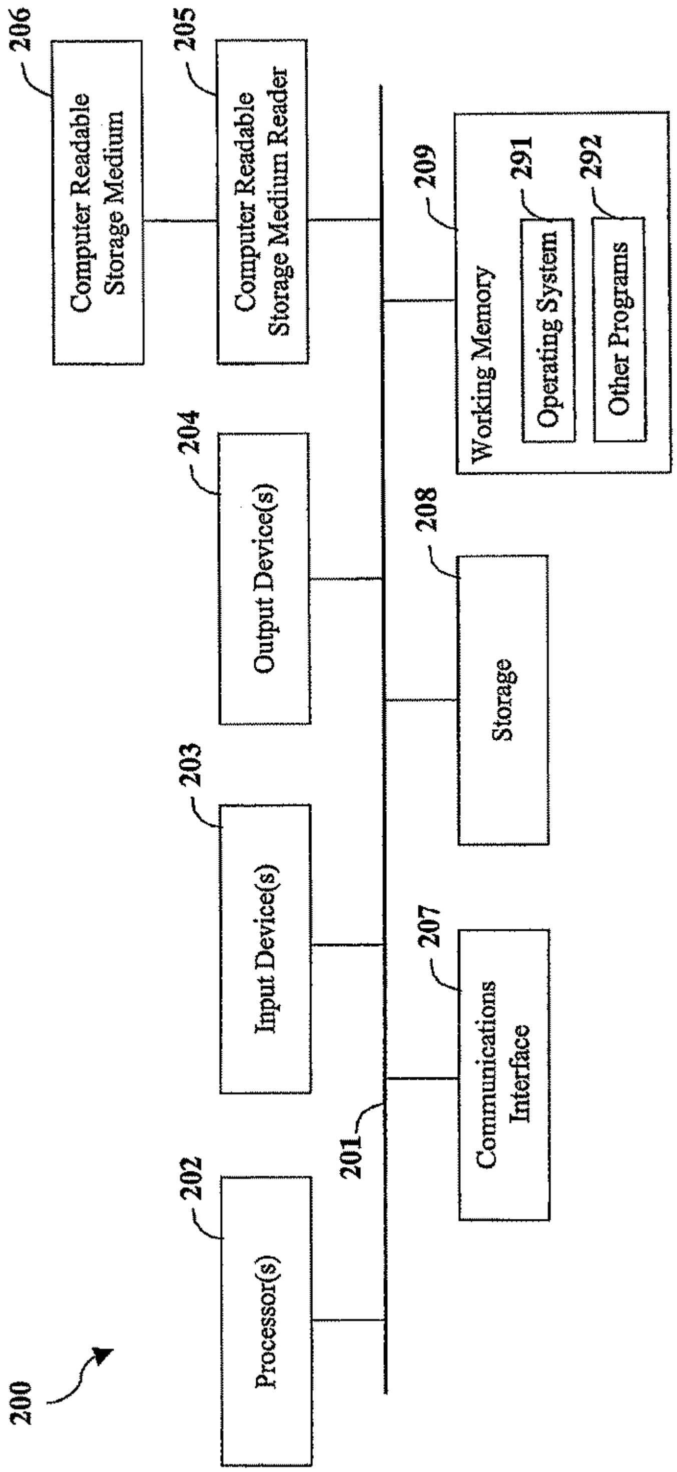

200
202
Processor(s)
201
203
Input Device(s)
204
Output Device(s)
205
Computer Readable Storage Medium Reader
206
Computer Readable Storage Medium
207
Communications Interface
208
Storage
209
Working Memory
291
Operating System
292
Other Programs

FIG. 2

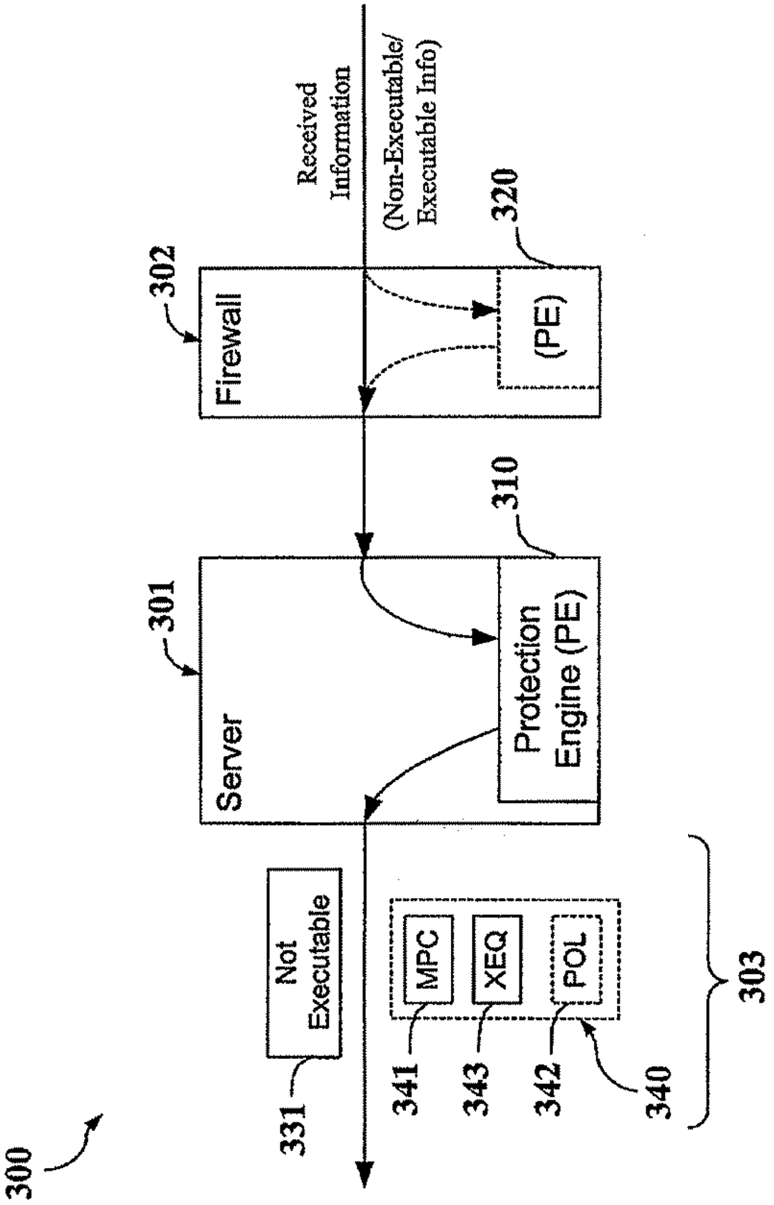

Received Information
(Non-Executable/ Executable Info)
302
Firewall
320
(PE)
310
301
Server
Protection Engine (PE)
Not Executable
331
MPC
XEQ
POL
341
343
342
340
303
300

FIG. 3

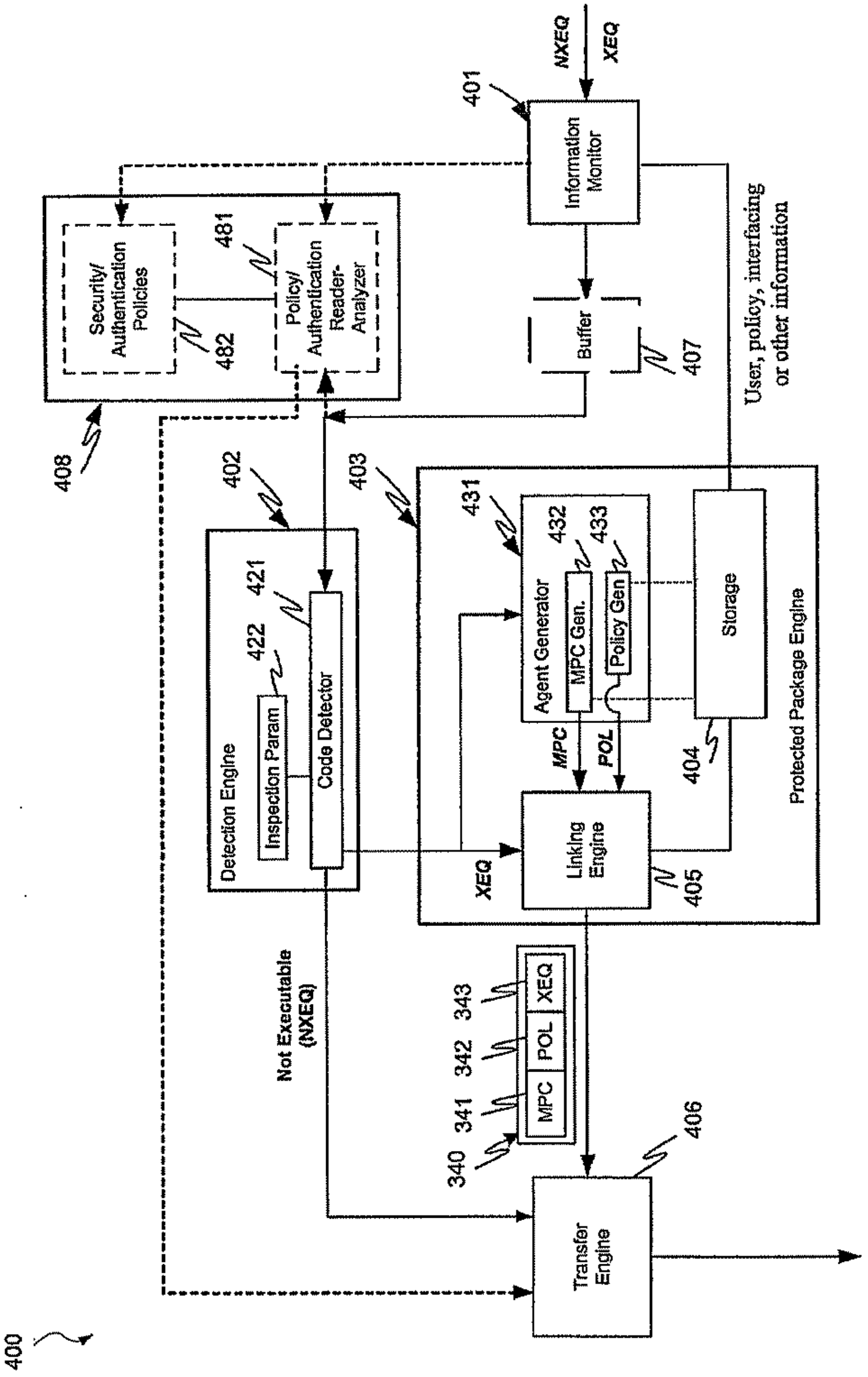

NXEQ
XEQ
401
Information Monitor
Security/ Authentication Policies
Policy/ Authentication Reader-Analyzer
481
482
Buffer
407
User, policy, interfacing or other information
408
402
403
431
432
433
Agent Generator
MPC Gen.
Policy Gen
Storage
404
Protected Package Engine
Detection Engine
Inspection Param
Code Detector
421
422
MPC
POL
XEQ
Linking Engine
405
Not Executable (NXEQ)
340
341
342
343
MPC
POL
XEQ
406
Transfer Engine
400

FIG. 4

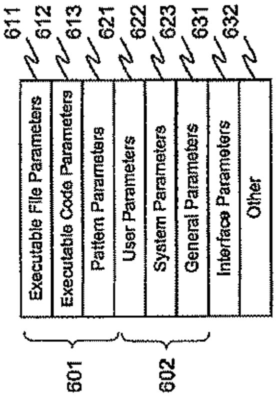


FIG. 6a

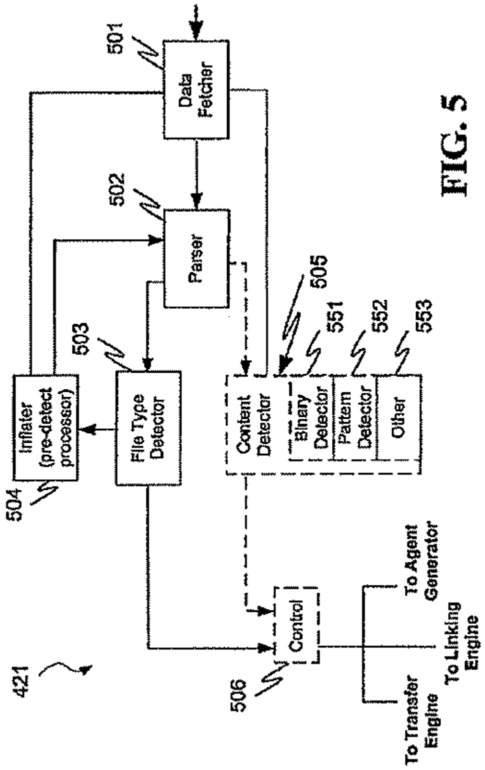


FIG. 5

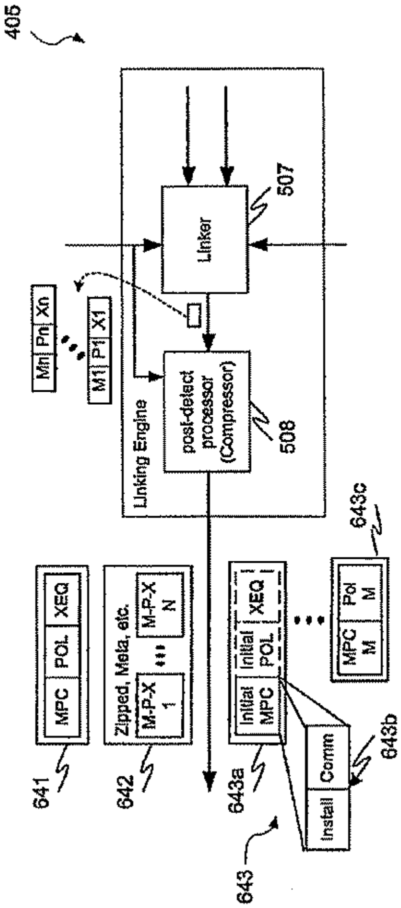


FIG. 6b

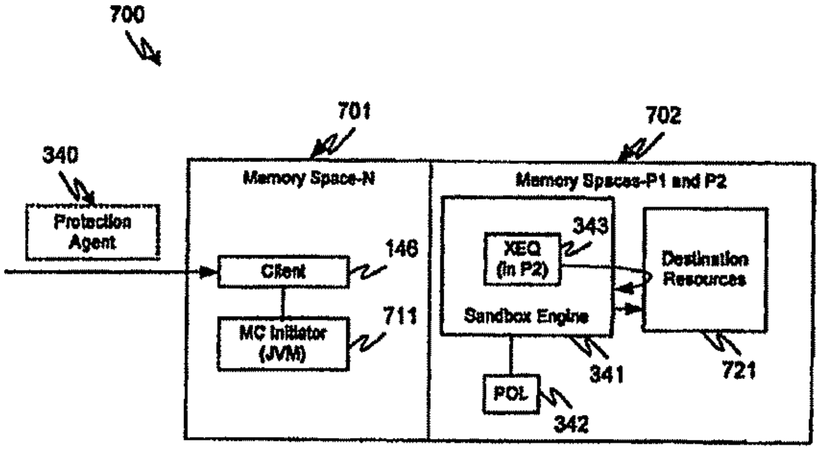


FIG. 7a

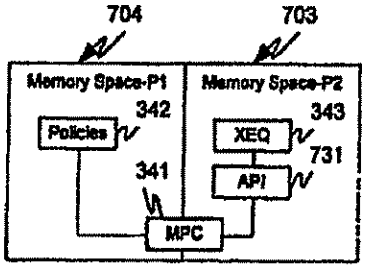


FIG. 7b

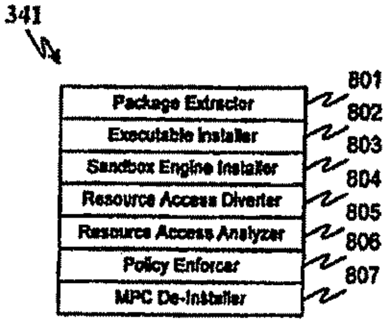


FIG. 8

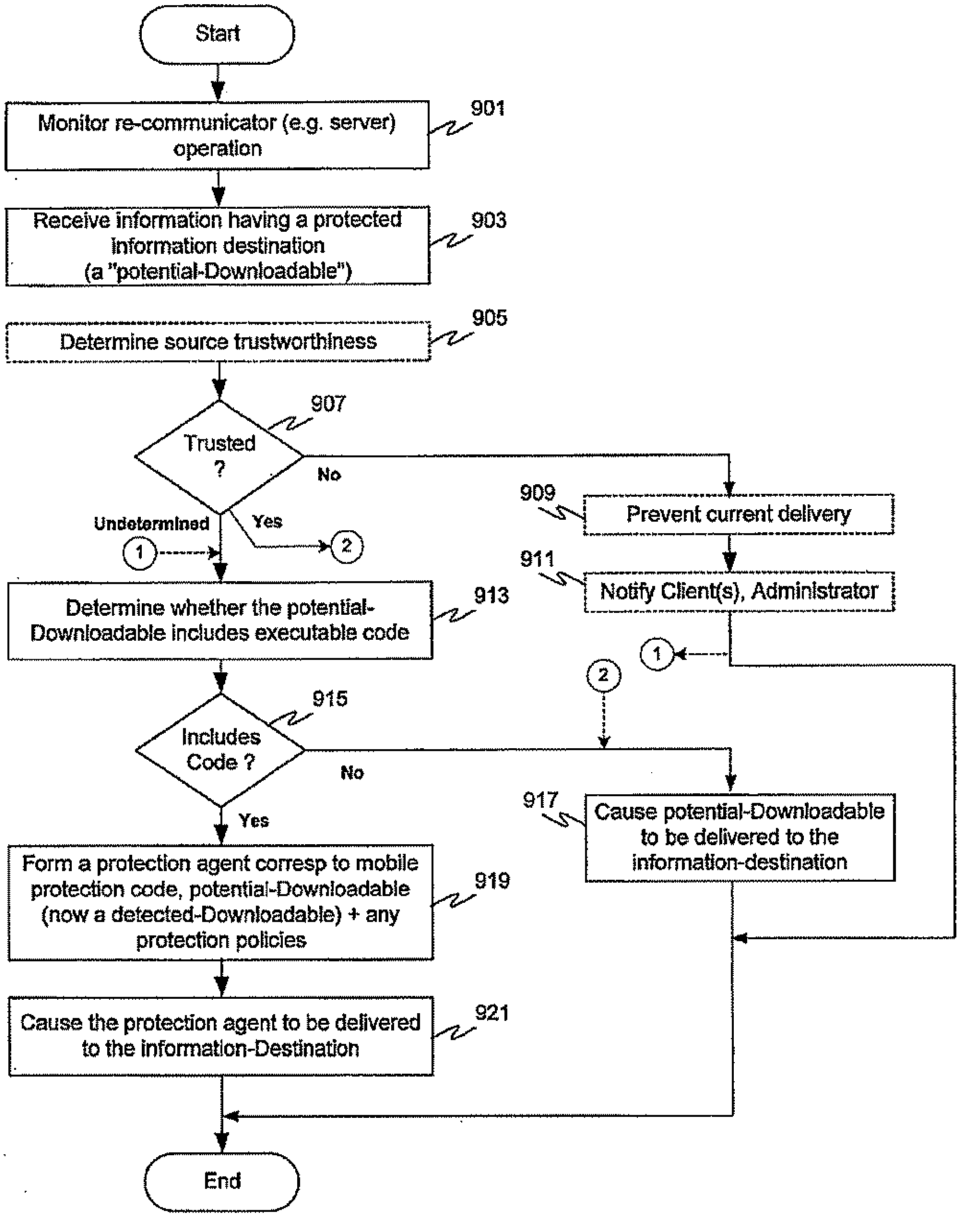

Start
Monitor re-communicator (e.g. server) operation
901
Receive information having a protected information destination (a "potential-Downloadable")
903
Determine source trustworthiness
905
Trusted ?
907
No
Undetermined
Yes
1
2
909
Prevent current delivery
911
Notify Client(s), Administrator
1
2
Determine whether the potential-Downloadable includes executable code
913
Includes Code ?
915
No
Yes
917
Cause potential-Downloadable to be delivered to the information-destination
Form a protection agent corresp to mobile protection code, potential-Downloadable (now a detected-Downloadable) + any protection policies
919
Cause the protection agent to be delivered to the information-Destination
921
End

FIG. 9

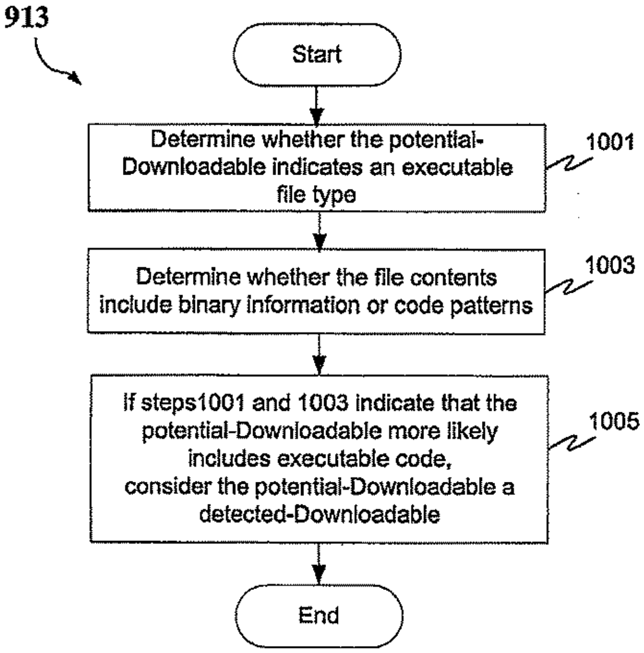

913
Start
Determine whether the potential-Downloadable indicates an executable file type
1001
Determine whether the file contents include binary information or code patterns
1003
If steps1001 and 1003 indicate that the potential-Downloadable more likely includes executable code, consider the potential-Downloadable a detected-Downloadable
1005
End

FIG. 10A

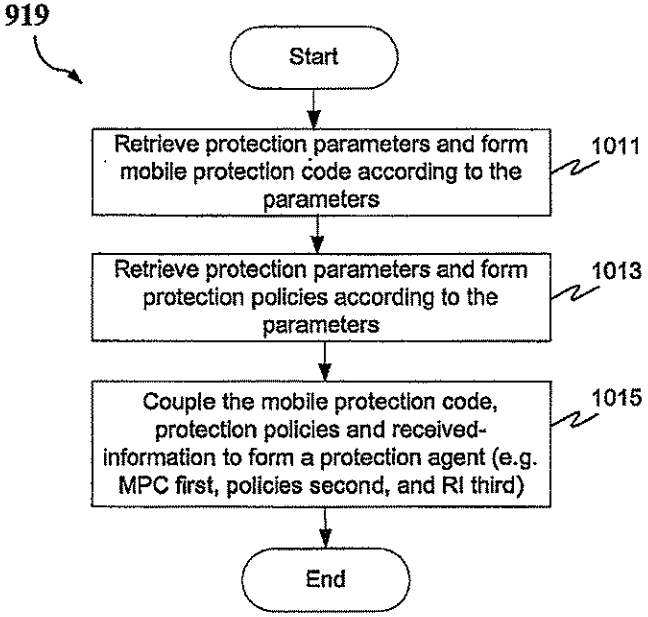

919
Start
Retrieve protection parameters and form mobile protection code according to the parameters
1011
Retrieve protection parameters and form protection policies according to the parameters
1013
Couple the mobile protection code, protection policies and received-information to form a protection agent (e.g. MPC first, policies second, and RI third)
1015
End

FIG. 10B

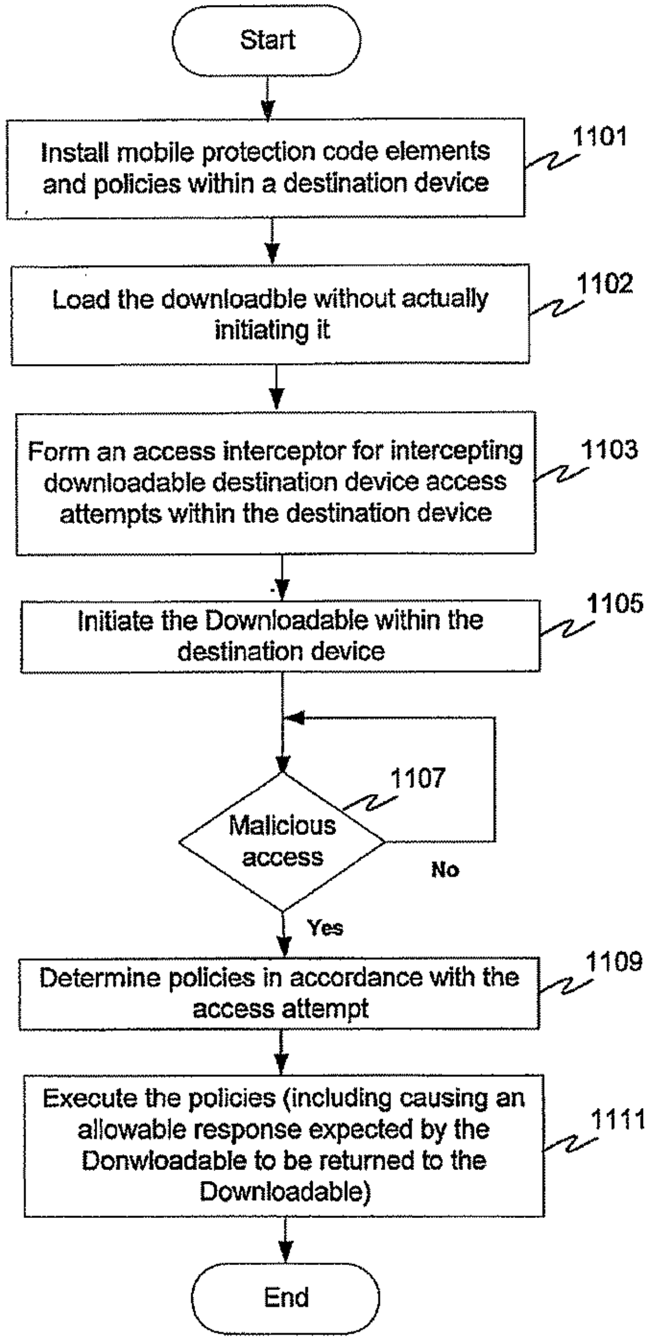

Start
Install mobile protection code elements and policies within a destination device
1101
Load the downloadble without actually initiating it
1102
Form an access interceptor for intercepting downloadable destination device access attempts within the destination device
1103
Initiate the Downloadable within the destination device
1105
1107
Malicious access
No
Yes
Determine policies in accordance with the access attempt
1109
Execute the policies (including causing an allowable response expected by the Donwloadable to be returned to the Downloadable)
1111
End

FIG. 11

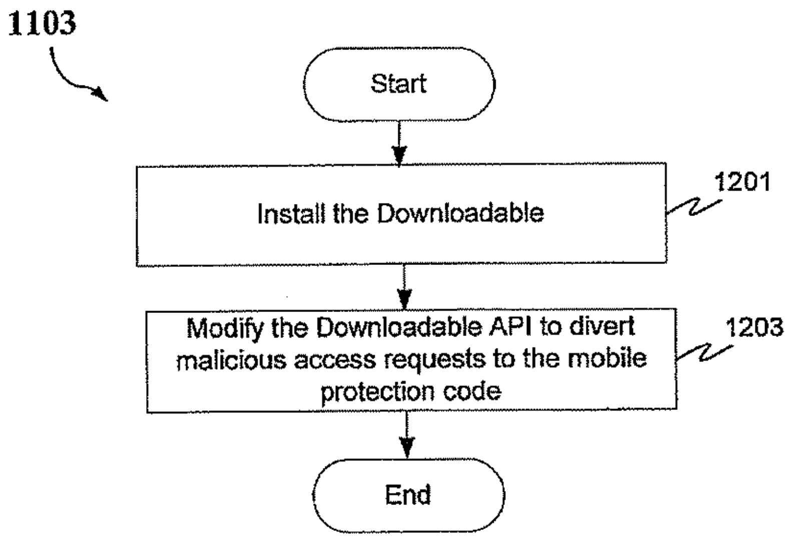

1103
Start
Install the Downloadable
1201
Modify the Downloadable API to divert malicious access requests to the mobile protection code
1203
End

FIG. 12a

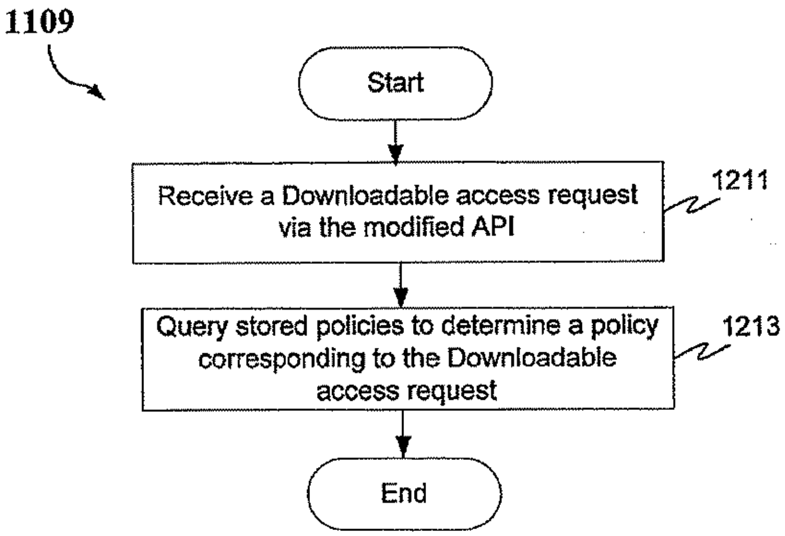

1109
Start
Receive a Downloadable access request via the modified API
1211
Query stored policies to determine a policy corresponding to the Downloadable access request
1213
End

FIG. 12b

MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS

PRIORITY REFERENCE TO RELATED APPLICATIONS

This application claims benefit of and hereby incorporates by reference provisional application Ser. No. 60/205,591, entitled "Computer Network Malicious Code Run-time Monitoring," filed on May 17, 2000 by inventors Nimrod Itzhak Vered, et al. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/539,667, now U.S. Pat. No. 6,804,780, entitled "System and Method for Protecting a Computer and a Network From Hostile Downloadables" filed on Mar. 30, 2000 by inventor Shlomo Touboul. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/551,302, now U.S. Pat. No. 6,480,962, entitled "System and Method for Protecting a Client During Runtime From Hostile Downloadables", filed on Apr. 18, 2000 by inventor Shlomo Touboul.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and methods for protecting network-connectable devices from undesirable downloadable operation.

2. Description of the Background Art

Advances in networking technology continue to impact an increasing number and diversity of users. The Internet, for example, already provides to expert, intermediate and even novice users the informational, product and service resources of over 100,000 interconnected networks owned by governments, universities, nonprofit groups, companies, etc. Unfortunately, particularly the Internet and other public networks have also become a major source of potentially system-fatal or otherwise damaging computer code commonly referred to as "viruses."

Efforts to forestall viruses from attacking networked computers have thus far met with only limited success at best. Typically, a virus protection program designed to identify and remove or protect against the initiating of known viruses is installed on a network firewall or individually networked computer. The program is then inevitably surmounted by some new virus that often causes damage to one or more computers. The damage is then assessed and, if isolated, the new virus is analyzed. A corresponding new virus protection program (or update thereof) is then developed and installed to combat the new virus, and the new program operates successfully until yet another new virus appears—and so on. Of course, damage has already typically been incurred.

To make matters worse, certain classes of viruses are not well recognized or understood, let alone protected against. It is observed by this inventor, for example, that Downloadable information comprising program code can include distributable components (e.g. Java™ applets and JavaScript scripts, ActiveX™ controls, Visual Basic, add-ins and/or others). It can also include, for example, application programs, Trojan horses, multiple compressed programs such as zip or meta files, among others. U.S. Pat. No. 5,983,348 to Shuang, however, teaches a protection system for protecting against only distributable components including "Java applets or ActiveX controls", and further does so using resource intensive and high bandwidth static Downloadable

content and operational analysis, and modification of the Downloadable component; Shuang further fails to detect or protect against additional program code included within a tested Downloadable. U.S. Pat. No. 5,974,549 to Golan teaches a protection system that further focuses only on protecting against ActiveX controls and not other distributable components, let alone other Downloadable types. U.S. Pat. No. 6,167,520 to Touboul enables more accurate protection than Shuang or Golan, but lacks the greater flexibility and efficiency taught herein, as do Shuang and Golan.

Accordingly, there remains a need for efficient, accurate and flexible protection of computers and other network connectable devices from malicious Downloadables.

SUMMARY OF THE INVENTION

The present invention provides protection systems and methods capable of protecting a personal computer ("PC") or other persistently or even intermittently network accessible devices or processes from harmful, undesirable, suspicious or other "malicious" operations that might otherwise be effectuated by remotely operable code. While enabling the capabilities of prior systems, the present invention is not nearly so limited, resource intensive or inflexible, and yet enables more reliable protection. For example, remotely operable code that is protectable against can include downloadable application programs, Trojan horses and program code groupings, as well as software "components", such as Java™ applets, ActiveX™ controls, JavaScript™/Visual Basic scripts, add-ins, etc., among others. Protection can also be provided in a distributed interactively, automatically or mixed configurable manner using protected client, server or other parameters, redirection, local/remote logging, etc., and other server/client based protection measures can also be separately and/or interoperably utilized, among other examples.

In one aspect, embodiments of the invention provide for determining, within one or more network "servers" (e.g. fireballs, resources, gateways, email relays or other devices/processes that are capable of receiving-and-transferring a Downloadable) whether received information includes executable code (and is a "Downloadable"). Embodiments also provide for delivering static, configurable and/or extensible remotely operable protection policies to a Downloadable-destination, more typically as a sandboxed package including the mobile protection code, downloadable policies and one or more received Downloadables. Further client-based or remote protection code/policies can also be utilized in a distributed manner. Embodiments also provide for causing the mobile protection code to be executed within a Downloadable-destination in a manner that enables various Downloadable operations to be detected, intercepted or further responded to via protection operations. Additional server/information-destination device security or other protection is also enabled, among still further aspects.

A protection engine according to an embodiment of the invention is operable within one or more network servers, firewalls or other network connectable information re-communicating devices (as are referred to herein summarily one or more "servers" or "re-communicators"). The protection engine includes an information monitor for monitoring information received by the server, and a code detection engine for determining whether the received information includes executable code. The protection engine also includes a packaging engine for causing a sandboxed package, typically including mobile protection code and downloadable protection policies to be sent to a Downloadable-destination in conjunction with the received information, if the received information is determined to be a Downloadable.

A sandboxed package according to an embodiment of the invention is receivable by and operable with a remote Downloadable-destination. The sandboxed package includes mobile protection code ("MPC") for causing one or more predetermined malicious operations or operation combinations of a Downloadable to be monitored or otherwise intercepted. The sandboxed package also includes protection policies (operable alone or in conjunction with further Downloadable-destination stored or received policies/MPCs) for causing one or more predetermined operations to be performed if one or more undesirable operations of the Downloadable is/are intercepted. The sandboxed package can also include a corresponding Downloadable and can provide for initiating the Downloadable in a protective "sandbox". The MPC/policies can further include a communicator for enabling further MPC/policy information or "modules" to be utilized and/or for event logging or other purposes.

A sandbox protection system according to an embodiment of the invention comprises an installer for enabling a received MPC to be executed within a Downloadable-destination (device/process) and further causing a Downloadable application program, distributable component or other received downloadable code to be received and installed within the Downloadable-destination. The protection system also includes a diverter for monitoring one or more operation attempts of the Downloadable, an operation analyzer for determining one or more responses to the attempts, and a security enforcer for effectuating responses to the monitored operations. The protection system can further include one or more security policies according to which one or more protection system elements are operable automatically (e.g. programmatically) or in conjunction with user intervention (e.g. as enabled by the security enforcer). The security policies can also be configurable/extensible in accordance with further downloadable and/or Downloadable-destination information.

A method according to an embodiment of the invention includes receiving downloadable information, determining whether the downloadable information includes executable code, and causing a mobile protection code and security policies to be communicated to a network client in conjunction with security policies and the downloadable information if the downloadable information is determined to include executable code. The determining can further provide multiple tests for detecting, alone or together, whether the downloadable information includes executable code.

A further method according to an embodiment of the invention includes forming a sandboxed package that includes mobile protection code ("MPC"), protection policies, and a received, detected-Downloadable, and causing the sandboxed package to be communicated to and installed by a receiving device or process ("user device") for responding to one or more malicious operation attempts by the detected-Downloadable from within the user device. The MPC/policies can further include a base "module" and a "communicator" for enabling further up/downloading of one or more further "modules" or other information (e.g. events, user/user device information, etc.).

Another method according to an embodiment of the invention includes installing, within a user device, received mobile protection code ("MPC") and protection policies in conjunction with the user device receiving a downloadable application program, component or other Downloadable(s). The method also includes determining, by the MPC, a resource access attempt by the Downloadable, and initiating, by the MPC, one or more predetermined operations corresponding to the attempt. (Predetermined operations can, for example, comprise initiating user, administrator, client, network or protection system determinable operations, including but not limited to modifying the Downloadable operation, extricating the Downloadable, notifying a user/another, maintaining a local/remote log, causing one or more MPCs/policies to be downloaded, etc.)

Advantageously, systems and methods according to embodiments of the invention enable potentially damaging, undesirable or otherwise malicious operations by even unknown mobile code to be detected, prevented, modified and/or otherwise protected against without modifying the mobile code. Such protection is further enabled in a manner that is capable of minimizing server and client resource requirements, does not require pre-installation of security code within a Downloadable-destination, and provides for client specific or generic and readily updateable security measures to be flexibly and efficiently implemented. Embodiments further provide for thwarting efforts to bypass security measures (e.g. by "hiding" undesirable operation causing information within apparently inert or otherwise "friendly" downloadable information) and/or dividing or combining security measures for even greater flexibility and/or efficiency.

Embodiments also provide for determining protection policies that can be downloaded and/or ascertained from other security information (e.g. browser settings, administrative policies, user input, uploaded information, etc.). Different actions in response to different Downloadable operations, clients, users and/or other criteria are also enabled, and embodiments provide for implementing other security measures, such as verifying a downloadable source, certification, authentication, etc. Appropriate action can also be accomplished automatically (e.g. programmatically) and/or in conjunction with alerting one or more users/administrators, utilizing user input, etc. Embodiments further enable desirable Downloadable operations to remain substantially unaffected, among other aspects.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1*a* is a block diagram illustrating a network system in accordance with an embodiment of the present invention;

FIG. 1*b* is a block diagram illustrating a network subsystem example in accordance with an embodiment of the invention;

FIG. 1*c* is a block diagram illustrating a further network subsystem example in accordance with an embodiment of the invention;

FIG. 2 is a block diagram illustrating a computer system in accordance with an embodiment of the invention;

FIG. 3 is a flow diagram broadly illustrating a protection system host according to an embodiment of the invention;

FIG. 4 is a block diagram illustrating a protection engine according to an embodiment of the invention;

FIG. 5 is a block diagram illustrating a content inspection engine according to an embodiment of the invention;

FIG. 6*a* is a block diagram illustrating protection engine parameters according to an embodiment of the invention;

FIG. 6*b* is a flow diagram illustrating a linking engine use in conjunction with ordinary, compressed and distributable sandbox package utilization, according to an embodiment of the invention;

FIG. 7*a* is a flow diagram illustrating a sandbox protection system operating within a destination system, according to an embodiment of the invention;

FIG. 7*b* is a block diagram illustrating memory allocation usable in conjunction with the protection system of FIG. 7*a*, according to an embodiment of the invention;

FIG. 8 is a block diagram illustrating a mobile protection code according to an embodiment of the invention;

FIG. 9 is a flowchart illustrating a server based protection method according to an embodiment of the invention;

FIG. 10*a* is a flowchart illustrating method for determining if a potential-Downloadable includes or is likely to include executable code, according to an embodiment of the invention;

FIG. 10*b* is a flowchart illustrating a method for forming a protection agent, according to an embodiment of the invention;

FIG. 11 is a flowchart illustrating a method for protecting a Downloadable destination according to an embodiment of the invention;

FIG. 12*a* is a flowchart illustrating a method for forming a Downloadable access interceptor according to an embodiment of the invention; and

FIG. 12*b* is a flowchart illustrating a method for implementing mobile protection policies according to an embodiment of the invention.

DETAILED DESCRIPTION

In providing malicious mobile code runtime monitoring systems and methods, embodiments of the invention enable actually or potentially undesirable operations of even unknown malicious code to be efficiently and flexibly avoided. Embodiments provide, within one or more "servers" (e.g. firewalls, resources, gateways, email relays or other information re-communicating devices), for receiving downloadable-information and detecting whether the downloadable-information includes one or more instances of executable code (e.g. as with a Trojan horse, zip/meta file etc.). Embodiments also provide for separately or interoperably conducting additional security measures within the server, within a Downloadable-destination of a detected-Downloadable, or both.

Embodiments further provide for causing mobile protection code ("MPC") and downloadable protection policies to be communicated to, installed and executed within one or more received information destinations in conjunction with a detected-Downloadable. Embodiments also provide, within an information-destination, for detecting malicious operations of the detected-Downloadable and causing responses thereto in accordance with the protection policies (which can correspond to one or more user, Downloadable, source, destination, or other parameters), or further downloaded or downloadable-destination based policies (which can also be configurable or extensible). (Note that the term "or", as used herein, is generally intended to mean "and/or" unless otherwise indicated.)

FIGS. 1*a* through 1*c* illustrate a computer network system 100 according to an embodiment of the invention. FIG. 1*a* broadly illustrates system 100, while FIGS. 1*b* and 1*c* illustrate exemplary protectable subsystem implementations corresponding with system 104 or 106 of FIG. 1*a*.

Beginning with FIG. 1*a*, computer network system 100 includes an external computer network 101, such as a Wide Area Network or "WAN" (e.g. the Internet), which is coupled to one or more network resource servers (summarily depicted as resource server-1 102 and resource server-N

103). Where external network 101 includes the Internet, resource servers 1-N (102, 103) might provide one or more resources including web pages, streaming media, transaction-facilitating information, program updates or other downloadable information, summarily depicted as resources 121, 131 and 132. Such information can also include more traditionally viewed "Downloadables" or "mobile code" (i.e. distributable components), as well as downloadable application programs or other further Downloadables, such as those that are discussed herein. (It will be appreciated that interconnected networks can also provide various other resources as well.)

Also coupled via external network 101 are subsystems 104–106. Subsystems 104–106 can, for example, include one or more servers, personal computers ("PCs"), smart appliances, personal information managers or other devices/processes that are at least temporarily or otherwise intermittently directly or indirectly connectable in a wired or wireless manner to external network 101 (e.g. using a dialup, DSL, cable modem, cellular connection, IR/RF, or various other suitable current or future connection alternatives). One or more of subsystems 104–106 might further operate as user devices that are connectable to external network 101 via an internet service provider ("ISP") or local area network ("LAN"), such as a corporate intranet, or home, portable device or smart appliance network, among other examples.

FIG. 1*a* also broadly illustrates how embodiments of the invention are capable of selectively, modifiably or extensibly providing protection to one or more determinable ones of networked subsystems 104–106 or elements thereof (not shown) against potentially harmful or other undesirable ("malicious") effects in conjunction with receiving downloadable information. "Protected" subsystem 104, for example, utilizes a protection in accordance with the teachings herein, while "unprotected" subsystem-N 105 employs no protection, and protected subsystem-M 106 might employ one or more protections including those according to the teachings herein, other protection, or some combination.

System 100 implementations are also capable of providing protection to redundant elements 107 of one or more of subsystems 104–106 that might be utilized, such as backups, failsafe elements, redundant networks, etc. Where included, such redundant elements are also similarly protectable in a separate, combined or coordinated manner using embodiments of the present invention either alone or in conjunction with other protection mechanisms. In such cases, protection can be similarly provided singly, as a composite of component operations or in a backup fashion. Care should, however, be exercised to avoid potential repeated protection engine execution corresponding to a single Downloadable; such "chaining" can cause a Downloadable to operate incorrectly or not at all, unless a subsequent detection engine is configured to recognize a prior packaging of the Downloadable.

FIGS. 1*b* and 1*c* further illustrate, by way of example, how protection systems according to embodiments of the invention can be utilized in conjunction with a wide variety of different system implementations. In the illustrated examples, system elements are generally configurable in a manner commonly referred to as a "client-server" configuration, as is typically utilized for accessing Internet and many other network resources. For clarity sake, a simple client-server configuration will be presumed unless otherwise indicated. It will be appreciated, however, that other configurations of interconnected elements might also be utilized (e.g. peer-peer, routers, proxy servers, networks, converters, gateways, services, network reconfiguring elements, etc.) in accordance with a particular application.

The FIG. **1***b* example shows how a suitable protected system **104***a* (which can correspond to subsystem-1 **104** or subsystem-M **106** of FIG. **1**) can include a protection-initiating host "server" or "re-communicator" (e.g. ISP server **140***a*), one or more user devices or "Downloadable-destinations" **145**, and zero or more redundant elements (which elements are summarily depicted as redundant client device/process **145***a*). In this example, ISP server **140***a* includes one or more email, Internet or other servers **141***a*, or other devices or processes capable of transferring or otherwise "re-communicating" downloadable information to user devices **145**. Server **141***a* further includes protection engine or "PE" **142***a*, which is capable of supplying mobile protection code ("MPC") and protection policies for execution by client devices **145**. One or more of user devices **145** can further include a respective one or more clients **146** for utilizing information received via server **140***a*, in accordance with which MPC and protection policies are operable to protect user devices **145** from detrimental, undesirable or otherwise "malicious" operations of downloadable information also received by user device **145**.

The FIG. **1***c* example shows how a further suitable protected system **104***b* can include, in addition to a "re-communicator", such as server **142***b*, a firewall **143***c* (e.g. as is typically the case with a corporate intranet and many existing or proposed home/smart networks.) In such cases, a server **141***b* or firewall **143** can operate as a suitable protection engine host. A protection engine can also be implemented in a more distributed manner among two or more protection engine host systems or host system elements, such as both of server **141***b* and firewall **143**, or in a more integrated manner, for example, as a standalone device. Redundant system or system protection elements can also be similarly provided in a more distributed or integrated manner (see above).

System **104***b* also includes internal network **144** and user devices **145**. User devices **145** further include a respective one or more clients **146** for utilizing information received via server **140***a*, in accordance with which the MPCs or protection policies are operable. (As in the previous example, one or more of user devices **145** can also include or correspond with similarly protectable redundant system elements, which are not shown.)

It will be appreciated that the configurations of FIGS. **1***a*–**1***c* are merely exemplary. Alternative embodiments might, for example, utilize other suitable connections, devices or processes. One or more devices can also be configurable to operate as a network server, firewall, smart router, a resource server servicing deliverable third-party/manufacturer postings, a user device operating as a firewall/server, or other information-suppliers or intermediaries (i.e. as a "re-communicator" or "server") for servicing one or more further interconnected devices or processes or interconnected levels of devices or processes. Thus, for example, a suitable protection engine host can include one or more devices or processes capable of providing or supporting the providing of mobile protection code or other protection consistent with the teachings herein. A suitable information-destination or "user device" can further include one or more devices or processes (such as email, browser or other clients) that are capable of receiving and initiating or otherwise hosting a mobile code execution.

FIG. **2** illustrates an exemplary computing system **200**, that can comprise one or more of the elements of FIGS. **1***a* through **1***c*. While other application-specific alternatives might be utilized, it will be presumed for clarity sake that system **100** elements (FIGS. **1***a*–*c*) are implemented in hardware, software or some combination by one or more processing systems consistent therewith, unless otherwise indicated.

Computer system **200** comprises elements coupled via communication channels (e.g. bus **201**) including one or more general or special purpose processors **202**, such as a Pentium® or Power PC®, digital signal processor ("DSP"), etc. System **200** elements also include one or more input devices **203** (such as a mouse, keyboard, microphone, pen, etc.), and one or more output devices **204**, such as a suitable display, speakers, actuators, etc., in accordance with a particular application.

System **200** also includes a computer readable storage media reader **205** coupled to a computer readable storage medium **206**, such as a storage/memory device or hard or removable storage/memory media; such devices or media are further indicated separately as storage device **208** and memory **209**, which can include hard disk variants, floppy/compact disk variants, digital versatile disk ("DVD") variants, smart cards, read only memory, random access memory, cache memory, etc., in accordance with a particular application. One or more suitable communication devices **207** can also be included, such as a modem, DSL, infrared or other suitable transceiver, etc. for providing inter-device communication directly or via one or more suitable private or public networks that can include but are not limited to those already discussed.

Working memory further includes operating system ("OS") elements and other programs, such as application programs, mobile code, data, etc. for implementing system **100** elements that might be stored or loaded therein during use. The particular OS can vary in accordance with a particular device, features or other aspects in accordance with a particular application (e.g. Windows, Mac, Linux, Unix or Palm OS variants, a proprietary OS, etc.). Various programming languages or other tools can also be utilized, such as C++, Java, Visual Basic, etc. As will be discussed, embodiments can also include a network client such as a browser or email client, e.g. as produced by Netscape, Microsoft or others, a mobile code executor such as an OS task manager, Java Virtual Machine ("JVM"), etc., and an application program interface ("API"), such as a Microsoft Windows or other suitable element in accordance with the teachings herein. (It will also become apparent that embodiments might also be implemented in conjunction with a resident application or combination of mobile code and resident application components.)

One or more system **200** elements can also be implemented in hardware, software or a suitable combination. When implemented in software (e.g. as an application program, object, downloadable, servlet, etc. in whole or part), a system **200** element can be communicated transitionally or more persistently from local or remote storage to memory (or cache memory, etc.) for execution, or another suitable mechanism can be utilized, and elements can be implemented in compiled or interpretive form. Input, intermediate or resulting data or functional elements can further reside more transitionally or more persistently in a storage media, cache or more persistent volatile or non-volatile memory, (e.g. storage device **207** or memory **208**) in accordance with a particular application.

FIG. **3** illustrates an interconnected re-communicator **300** generally consistent with system **140***b* of FIG. **1**, according to an embodiment of the invention. As with system **140***b*, system **300** includes a server **301**, and can also include a firewall 302. In this implementation, however, either server 301 or firewall 302 (if a firewall is used) can further include a protection engine (310 or 320 respectively). Thus, for example, an included firewall can process received information in a conventional manner, the results of which can be further processed by protection engine 310 of server 301, or information processed by protection engine 320 of an included firewall 302 can be processed in a conventional manner by server 301. (For clarity sake, a server including a singular protection engine will be presumed, with or without a firewall, for the remainder of the discussion unless otherwise indicated. Note, however, that other embodiments consistent with the teachings herein might also be utilized.)

FIG. 3 also shows how information received by server 301 (or firewall 302) can include non-executable information, executable information or a combination of non-executable and one or more executable code portions (e.g. so-called Trojan horses that include a hostile Downloadable within a friendly one, combined, compressed or otherwise encoded files, etc.). Particularly such combinations will likely remain undetected by a firewall or other more conventional protection systems. Thus, for convenience, received information will also be referred to as a "potential-Downloadable", and received information found to include executable code will be referred to as a "Downloadable" or equivalently as a "detected-Downloadable" (regardless of whether the executable code includes one or more application programs, distributable "components" such as Java, ActiveX, add-in, etc.).

Protection engine 310 provides for detecting whether received potential-Downloadables include executable code, and upon such detection, for causing mobile protection code ("MPC") to be transferred to a device that is a destination of the potential-Downloadable (or "Downloadable-destination"). Protection engine 310 can also provide protection policies in conjunction with the MPC (or thereafter as well), which MPC/policies can be automatically (e.g. programmatically) or interactively configurable in accordance user, administrator, downloadable source, destination, operation, type or various other parameters alone or in combination (see below). Protection engine 310 can also provide or operate separately or interoperably in conjunction with one or more of certification, authentication, downloadable tagging, source checking, verification, logging, diverting or other protection services via the MPC, policies, other local/remote server or destination processing, etc. (e.g. which can also include protection mechanisms taught by the above-noted prior applications; see FIG. 4).

Operationally, protection engine 310 of server 301 monitors information received by server 301 and determines whether the received information is deliverable to a protected destination, e.g. using a suitable monitor/data transfer mechanism and comparing a destination-address of the received information to a protected destination set, such as a protected destinations list, array, database, etc. (All deliverable information or one or more subsets thereof might also be monitored.) Protection engine 310 further analyzes the potential-Downloadable and determines whether the potential-Downloadable includes executable code. If not, protection engine 310 enables the not executable potential-Downloadable 331 to be delivered to its destination in an unaffected manner.

In conjunction with determining that the potential-Downloadable is a detected-Downloadable, protection engine 310 also causes mobile protection code or "MPC" 341 to be communicated to the Downloadable-destination of the Downloadable, more suitably in conjunction with the

detected-Downloadable 343 (see below). Protection engine 310 further causes downloadable protection policies 342 to be delivered to the Downloadable-destination, again more suitably in conjunction with the detected-Downloadable. Protection policies 342 provide parameters (or can additionally or alternatively provide additional mobile code) according to which the MPC is capable of determining or providing applicable protection to a Downloadable-destination against malicious Downloadable operations.

(One or more "checked", tag, source, destination, type, detection or other security result indicators, which are not shown, can also be provided as corresponding to determined non-Downloadables or Downloadables, e.g. for testing, logging, further processing, further identification tagging or other purposes in accordance with a particular application.)

Further MPCs, protection policies or other information are also deliverable to a the same or another destination, for example, in accordance with communication by an MPC/protection policies already delivered to a downloadable-destination. Initial or subsequent MPCs/policies can further be selected or configured in accordance with a Downloadable-destination indicated by the detected-Downloadable, destination-user or administrative information, or other information providable to protection engine 310 by a user, administrator, user system, user system examination by a communicated MPC, etc. (Thus, for example, an initial MPC/policies can also be initially provided that are operable with or optimized for more efficient operation with different Downloadable-destinations or destination capabilities.)

While integrated protection constraints within the MPC might also be utilized, providing separate protection policies has been found to be more efficient, for example, by enabling more specific protection constraints to be more easily updated in conjunction with detected-Downloadable specifics, post-download improvements, testing, etc. Separate policies can further be more efficiently provided (e.g. selected, modified, instantiated, etc.) with or separately from an MPC, or in accordance with the requirements of a particular user, device, system, administration, later improvement, etc., as might also be provided to protection engine 310 (e.g. via user/MPC uploading, querying, parsing a Downloadable, or other suitable mechanism implemented by one or more servers or Downloadable-destinations).

(It will also become apparent that performing executable code detection and communicating to a downloadable-Destination an MPC and any applicable policies as separate from a detected-Downloadable is more accurate and far less resource intensive than, for example, performing content and operation scanning, modifying a Downloadable, or providing completely Downloadable-destination based security.)

System 300 enables a single or extensible base-MPC to be provided, in anticipation or upon receipt of a first Downloadable, that is utilized thereafter to provide protection of one or more Downloadable-destinations. It is found, however, that providing an MPC upon each detection of a Downloadable (which is also enabled) can provide a desirable combination of configurability of the MPC/policies and lessened need for management (e.g. given potentially changing user/destination needs, enabling testing, etc.).

Providing an MPC upon each detection of a Downloadable also facilitates a lessened demand on destination resources, e.g. since information-destination resources used in executing the MPC/policies can be re-allocated following such use. Such alternatives can also be selectively, modifiably or extensibly provided (or further in accordance with other application-specific factors that might also apply.)

Thus, for example, a base-MPC or base-policies might be provided to a user device that is/are extensible via additionally downloadable "modules" upon server 301 detection of a Downloadable deliverable to the same user device, among other alternatives.

In accordance with a further aspect of the invention, it is found that improved efficiency can also be achieved by causing the MPC to be executed within a Downloadable-destination in conjunction with, and further, prior to initiation of the detected Downloadable. One mechanism that provides for greater compatibility and efficiency in conjunction with conventional client-based Downloadable execution is for a protection engine to form a sandboxed package 340 including MPC 341, the detected-Downloadable 343 and any policies 342. For example, where the Downloadable is a binary executable to be executed by an operating system, protection engine 310 forms a protected package by concatenating, within sandboxed package 340, MPC 341 for delivery to a Downloadable-destination first, followed by protection policies 342 and Downloadable 343. (Concatenation or techniques consistent therewith can also be utilized for providing a protecting package corresponding to a Java applet for execution by a JVM of a Downloadable-destination, or with regard to ActiveX controls, add-ins or other distributable components, etc.)

The above concatenation or other suitable processing will result in the following. Upon receipt of sandboxed package 340 by a compatible browser, email or other destination-client and activating of the package by a user or the destination-client, the operating system (or a suitable responsively initiated distributed component host) will attempt to initiate sandboxed package 340 as a single Downloadable. Such processing will, however, result in initiating the MPC 341 and—in accordance with further aspects of the invention—the MPC will initiate the Downloadable in a protected manner, further in accordance with any applicable included or further downloaded protection policies 342. (While system 300 is also capable of ascertaining protection policies stored at a Downloadable-destination, e.g. by poll, query, etc. of available destination information, including at least initial policies within a suitable protecting package is found to avoid associated security concerns or inefficiencies.)

Turning to FIG. 4, a protection engine 400 generally consistent with protection engine 310 (or 320) of FIG. 3 is illustrated in accordance with an embodiment of the invention. Protection engine 400 comprises information monitor 401, detection engine 402, and protected packaging engine 403, which further includes agent generator 431, storage 404, linking engine 405, and transfer engine 406. Protection engine 400 can also include a buffer 407, for temporarily storing a received potential-Downloadable, or one or more systems for conducting additional authentication, certification, verification or other security processing (e.g. summarily depicted as security system 408) Protection engine 400 can further provide for selectively re-directing, further directing, logging, etc. of a potential/detected Downloadable or information corresponding thereto in conjunction with detection, other security, etc., in accordance with a particular application.

(Note that FIG. 4, as with other figures included herein, also depicts exemplary signal flow arrows; such arrows are provided to facilitate discussion, and should not be construed as exclusive or otherwise limiting.)

Information monitor 401 monitors potential-Downloadables received by a host server and provides the information via buffer 407 to detection engine 402 or to other system 400 elements. Information monitor 401 can be configured to monitor host server download operations in conjunction with a user or a user-device that has logged-on to the server, or to receive information via a server operation hook, servlet, communication channel or other suitable mechanism.

Information monitor 401 can also provide for transferring, to storage 404 or other protection engine elements, configuration information including, for example, user, MPC, protection policy, interfacing or other configuration information (e.g. see FIG. 6). Such configuration information monitoring can be conducted in accordance with a user/device logging onto or otherwise accessing a host server, via one or more of configuration operations, using an applet to acquire such information from or for a particular user, device or devices, via MPC/policy polling of a user device, or via other suitable mechanisms.

Detection engine 402 includes code detector 421, which receives a potential-Downloadable and determines, more suitably in conjunction with inspection parameters 422, whether the potential-Downloadable includes executable code and is thus a "detected-Downloadable". (Code detector 421 can also include detection processors for performing file decompression or other "decoding", or such detection-facilitating processing as decryption, utilization/support of security system 408, etc. in accordance with a particular application.)

Detection engine 402 further transfers a detected-downloadable ("XEQ") to protected packaging engine 403 along with indicators of such detection, or a determined non-executable ("NXEQ") to transfer engine 406. (Inspection parameters 422 enable analysis criteria to be readily updated or varied, for example, in accordance with particular source, destination or other potential Downloadable impacting parameters, and are discussed in greater detail with reference to FIG. 5). Detection engine 402 can also provide indicators for delivery of initial and further MPCs/policies, for example, prior to or in conjunction with detecting a Downloadable and further upon receipt of an indicator from an already downloaded MPC/policy. A downloaded MPC/policy can further remain resident at a user device with further modules downloaded upon or even after delivery of a sandboxed package. Such distribution can also be provided in a configurable manner, such that delivery of a complete package or partial packages are automatically or interactively determinable in accordance with user/administrative preferences/policies, among other examples.

Packaging engine 403 provides for generating mobile protection code and protection policies, and for causing delivery thereof (typically with a detected-Downloadable) to a Downloadable-destination for protecting the Downloadable-destination against malicious operation attempts by the detected Downloadable. In this example, packaging engine 403 includes agent generator 431, storage 404 and linking engine 405.

Agent generator 431 includes an MPC generator 432 and a protection policy generator 433 for "generating" an MPC and a protection policy (or set of policies) respectively upon receiving one or more "generate MPC/policy" indicators from detection engine 402, indicating that a potential-Downloadable is a detected-Downloadable. MPC generator 432 and protection policy generator 433 provide for generating MPCs and protection policies respectively in accordance with parameters retrieved from storage 404. Agent generator 431 is further capable of providing multiple MPCs/policies, for example, the same or different MPCs/policies in accordance with protecting ones of multiple executables within a zip file, or for providing initial MPCs/policies and then further MPCs/policies or MPC/policy "modules" as initiated by further indicators such as given above, via an indicator of an already downloaded MPC/policy or via other suitable mechanisms. (It will be appreciated that pre-constructed MPCs/policies or other processing can also be utilized, e.g. via retrieval from storage **404**, but with a potential decrease in flexibility.)

MPC generator **432** and protection policy generator **433** are further configurable. Thus, for example, more generic MPCs/policies can be provided to all or a grouping of serviced destination-devices (e.g. in accordance with a similarly configured/administered intranet), or different MPCs/policies that can be configured in accordance with one or more of user, network administration, Downloadable-destination or other parameters (e.g. see FIG. **6**). As will become apparent, a resulting MPC provides an operational interface to a destination device/process. Thus, a high degree of flexibility and efficiency is enabled in providing such an operational interface within different or differently configurable user devices/processes or other constraints.

Such configurability further enables particular policies to be utilized in accordance with a particular application (e.g. particular system uses, access limitations, user interaction, treating application programs or Java components from a particular known source one way and unknown source ActiveX components, or other considerations). Agent generator **431** further transfers a resulting MPC and protection policy pair to linking engine **405**.

Linking engine **405** provides for forming from received component elements (see above) a sandboxed package that can include one or more initial or complete MPCs and applicable protection policies, and a Downloadable, such that the sandboxed package will protect a receiving Downloadable-destination from malicious operation by the Downloadable. Linking engine **405** is implementable in a static or configurable manner in accordance, for example, with characteristics of a particular user device/process stored intermittently or more persistently in storage **404**. Linking engine **405** can also provide for restoring a Downloadable, such as a compressed, encrypted or otherwise encoded file that has been decompressed, decrypted or otherwise decoded via detection processing (e.g. see FIG. **6***b*).

It is discovered, for example, that the manner in which the Windows OS initiates a binary executable or an ActiveX control can be utilized to enable protected initiation of a detected-Downloadable. Linking engine **405** is, for example, configurable to form, for an ordinary single-executable Downloadable (e.g. an application program, applet, etc.) a sandboxed package **340** as a concatenation of ordered elements including an MPC **341**, applicable policies **342** and the Downloadable or "XEQ" **343** (e.g. see FIG. **4**).

Linking engine **405** is also configurable to form, for a Downloadable received by a server as a compressed single or multiple-executable Downloadable such as a zipped or meta file, a protecting package **340** including one or more MPCs, applicable policies and the one or more included executables of the Downloadable. For example, a sandboxed package can be formed in which a single MPC and policies precede and thus will affect all such executables as a result of inflating and installation. An MPC and applicable policies can also, for example, precede each executable, such that each executable will be separately sandboxed in the same or a different manner according to MPC/policy configuration (see above) upon inflation and installation. (See also FIGS. **5** and **6**)

Linking engine is also configurable to form an initial MPC, MPC-policy or sandboxed package (e.g. prior to upon receipt of a downloadable) or an additional MPC, MPC-policy or sandboxed package (e.g. upon or following receipt of a downloadable), such that suitable MPCs/policies can be provided to a Downloadable-destination or other destination in a more distributed manner. In this way, requisite bandwidth or destination resources can be minimized (via two or more smaller packages) in compromise with latency or other considerations raised by the additional required communication.

A configurable linking engine can also be utilized in accordance with other requirements of particular devices/processes, further or different elements or other permutations in accordance with the teachings herein. (It might, for example be desirable to modify the ordering of elements, to provide one or more elements separately, to provide additional information, such as a header, etc., or perform other processing in accordance with a particular device, protocol or other application considerations.)

Policy/authentication reader-analyzer **481** summarily depicts other protection mechanisms that might be utilized in conjunction with Downloadable detection, such as already discussed, and that can further be configurable to operate in accordance with policies or parameters (summarily depicted by security/authentication policies **482**). Integration of such further protection in the depicted configuration, for example, enables a potential-Downloadable from a known unfriendly source, a source failing authentication or a provided-source that is confirmed to be fictitious to be summarily discarded, otherwise blocked, flagged, etc. (with or without further processing). Conversely, a potential-Downloadable from a known friendly source (or one confirmed as such) can be transferred with or without further processing in accordance with particular application considerations. (Other configurations including pre or post Downloadable detection mechanisms might also be utilized.)

Finally, transfer engine **406** provides for receiving and causing linking engine **405** (or other protection) results to be transferred to a destination user device/process. As depicted, transfer engine **406** is configured to receive and transfer a Downloadable, a determined non-executable or a sandboxed package. However, transfer engine **406** can also be provided in a more configurable manner, such as was already discussed for other system **400** elements. (Any one or more of system **400** elements might be configurably implemented in accordance with a particular application.) Transfer engine **406** can perform such transfer, for example, by adding the information to a server transfer queue (not shown) or utilizing another suitable method.

Turning to FIG. **5** with reference to FIG. **4**, a code detector **421** example is illustrated in accordance with an embodiment of the invention. As shown, code detector **421** includes data fetcher **501**, parser **502**, file-type detector **503**, inflator **504** and control **506**; other depicted elements. While implementable and potentially useful in certain instances, are found to require substantial overhead, to be less accurate in certain instances (see above) and are not utilized in a present implementation; these will be discussed separately below. Code detector elements are further configurable in accordance with stored parameters retrievable by data fetcher **501**. (A coupling between data fetcher **501** and control **506** has been removed for clarity sake.)

Data fetcher **501** provides for retrieving a potential-Downloadable or portions thereof stored in buffer **407** or parameters from storage **404**, and communicates such information or parameters to parser **502**. Parser **502** receives a potential-Downloadable or portions thereof from data fetcher **501** and isolates potential-Downloadable elements, such as file headers, source, destination, certificates, etc. for use by further processing elements.

File type detector **502** receives and determines whether the potential-Downloadable (likely) is or includes an executable file type. File-reader **502** can, for example, be configured to analyze a received potential-Downloadable for a file header, which is typically included in accordance with conventional data transfer protocols, such as a portable executable or standard ".exe" file format for Windows OS application programs, a Java class header for Java applets, and so on for other applications, distributed components, etc. "Zipped", meta or other compressed files, which might include one or more executables, also typically provide standard single or multi-level headers that can be read and used to identify included executable code (or other included information types). File type detector **502** is also configurable for analyzing potential-Downloadables for all potential file type delimiters or a more limited subset of potential file type delimiters (e.g. ".exe" or ".com" in conjunction with a DOS or Microsoft Windows OS Downloadable-destination).

Known file type delimiters can, for example, be stored in a more temporary or more persistent storage (e.g. storage **404** of FIG. **4**) which file type detector **502** can compare to a received potential-Downloadable. (Such delimiters can thus also be updated in storage **404** as a new file type delimiter is provided, or a more limited subset of delimiters can also be utilized in accordance with a particular Downloadable-destination or other considerations of a particular application.) File type detector **502** further transfers to controller **506** a detected file type indicator indicating that the potential-Downloadable includes or does not include (i.e. or likely include) an executable file type.

In this example, the aforementioned detection processor is also included as predetection processor or, more particularly, a configurable file inflator **504**. File inflator **504** provides for opening or "inflating" compressed files in accordance with a compressed file type received from file type detector **503** and corresponding file opening parameters received from data fetcher **501**. Where a compressed file (e.g. a meta file) includes nested file type information not otherwise reliably provided in an overall file header or other information, inflator **504** returns such information to parser **502**. File inflator **504** also provides any now-accessible included executables to control **506** where one or more included files are to be separately packaged with an MPC or policies.

Control **506**, in this example, operates in accordance with stored parameters and provides for routing detected non-Downloadables or Downloadables and control information, and for conducting the aforementioned distributed downloading of packages to Downloadable-destinations. In the case of a non-Downloadable, for example, control **506** sends the non-Downloadable to transfer engine **406** (FIG. **4**) along with any indicators that might apply. For an ordinary single-executable Downloadable, control **506** sends control information to agent generator **431** and the Downloadable to linking engine **405** along with any other applicable indicators (see **641** of FIG. **6*b***). Control **506** similarly handles a compressed single-executable Downloadable or a multiple downloadable to be protected using a single sandboxed package. For a multiple-executable Downloadable, control **506** sends control information for each corresponding executable to agent generator agent generator **431**, and sends the executable to linking engine **405** along with controls and any applicable indicators, as in **643*b*** of FIG. **6*b***. (The above assumes, however, that distributed downloading is not utilized; when used—according to applicable parameters—control **506** also operates in accordance with the following.)

Control **506** conducts distributed protection (e.g. distributed packaging) by providing control signals to agent generator **431**, linking engine **405** and transfer engine **406**. In the present example, control **506** initially sends controls to agent generator **431** and linking engine **405** (FIG. **4**) causing agent generator to generate an initial MPC and initial policies, and sends control and a detected-Downloadable to linking engine **405**. Linking engine **405** forms an initial sandboxed package, which transfer engine causes (in conjunction with further controls) to be downloaded to the Downloadable destination (**643*a*** of FIG. **6*b***). An initial MPC within the sandboxed package includes an installer and a communicator and performs installation as indicated below. The initial MPC also communicates via the communicator controls to control **506** (FIG. **5**) in response to which control **506** similarly causes generation of MPC-M and policy-M modules **643*c***, which linking engine **405** links and transfer engine **406** causes to be sent to the Downloadable destination, and so on for any further such modules.

(It will be appreciated, however, that an initial package might be otherwise configured or sent prior to receipt of a Downloadable in accordance with configuration parameters or user interaction. Information can also be sent to other user devices, such as that of an administrator. Further MPCs/policies might also be coordinated by control **506** or other elements, or other suitable mechanisms might be utilized in accordance with the teachings herein.)

Regarding the remaining detection engine elements illustrated in FIG. **5**, where content analysis is utilized, parser **502** can also provide a Downloadable or portions thereof to content detector **505**. Content detector **505** can then provide one or more content analyses. Binary detector **551**, for example, performs detection of binary information; pattern detector **552** further analyzes the Downloadable for patterns indicating executable code, or other detectors can also be utilized. Analysis results therefrom can be used in an absolute manner, where a first testing result indicating executable code confirms Downloadable detection, which result is then sent to control **506**. Alternatively, however, composite results from such analyses can also be sent to control **506** for evaluation. Control **506** can further conduct such evaluation in a summary manner (determining whether a Downloadable is detected according to a majority or minimum number of indicators), or based on a weighting of different analysis results. Operation then continues as indicated above. (Such analysis can also be conducted in accordance with aspects of a destination user device or other parameters.)

FIG. **6*a*** illustrates more specific examples of indicators/parameters and known (or "knowledge base") elements that can be utilized to facilitate the above-discussed system **400** configurability and detection. For clarity sake, indicators, parameters and knowledge base elements are combined as indicated "parameters." It will be appreciated, however, that the particular parameters utilized can differ in accordance with a particular application, and indicators, parameters or known elements, where utilized, can vary and need not correspond exactly with one another. Any suitable explicit or referencing list, database or other storage structure(s) or storage structure configuration(s) can also be utilized to implement a suitable user/device based protection scheme, such as in the above examples, or other desired protection schema.

Executable parameters **601** comprise, in accordance with the above examples, executable file type parameters **611**, executable code parameters **612** and code pattern parameters **613** (including known executable file type indicators, header/code indicators and patterns respectively, where code patterns are utilized). Use parameters **602** further comprise user parameters **621**, system parameters **622** and general parameters **623** corresponding to one or more users, user classifications, user-system correspondences or destination system, device or processes, etc. (e.g. for generating corresponding MPCs/policies, providing other protection, etc.). The remaining parameters include interface parameters **631** for providing MPC/policy (or further) configurability in accordance with a particular device or for enabling communication with a device user (see below), and other parameters **632**.

FIG. **6***b* illustrates a linking engine **405** according to an embodiment of the invention. As already discussed, linking engine **405** includes a linker for combining MPCs, policies or agents via concatination or other suitable processing in accordance with an OS, JVM or other host executor or other applicable factors that might apply. Linking engine **405** also includes the aforementioned post-detection processor which, in this example, comprises a compressor **508**. As noted, compressor **508** receives linked elements from linker **507** and, where a potential-Downloadable corresponds to a compressed file that was inflated during detection, re-forms the compressed file. (Known file information can be provided via configuration parameters, substantially reversal of inflating or another suitable method.) Encryption or other post-detection processing can also be conducted by linking engine **508**.

FIGS. **7***a*, **7***b* and **8** illustrate a "sandbox protection" system, as operable within a receiving destination-device, according to an embodiment of the invention.

Beginning with FIG. **7***a*, a client **146** receiving sandbox package **340** will "recognize" sandbox package **340** as a (mobile) executable and cause a mobile code installer **711** (e.g. an OS loader, JVM, etc.) to be initiated. Mobile code installer **711** will also recognize sandbox package **340** as an executable and will attempt to initiate sandbox package **340** at its "beginning." Protection engine **400** processing corresponding to destination **700** use of a such a loader, however, will have resulted in the "beginning" of sandbox package **340** as corresponding to the beginning of MPC **341**, as noted with regard to the above FIG. **4** example.

Such protection engine processing will therefore cause a mobile code installer (e.g. OS loader **711**, for clarity sake) to initiate MPC **341**. In other cases, other processing might also be utilized for causing such initiation or further protection system operation. Protection engine processing also enables MPC **341** to effectively form a protection "sandbox" around Downloadable (e.g. detected-Downloadable or "XEQ") **343**, to monitor Downloadable **343**, intercept determinable Downloadable **343** operation (such as attempted accesses of Downloadable **343** to destination resources) and, if "malicious", to cause one or more other operations to occur (e.g. providing an alert, offloading the Downloadable, offloading the MPC, providing only limited resource access, possibly in a particular address space or with regard to a particularly "safe" resource or resource operation, etc.).

MPC **341**, in the present OS example, executes MPC element installation and installs any policies, causing MPC **341** and protection policies **342** to be loaded into a first memory space, P1. MPC **341** then initiates loading of Downloadable **343**. Such Downloadable initiation causes OS loader **711** to load Downloadable **343** into a further working memory space-P2 **703** along with an API import table ("IAT") **731** for providing Downloadable **631** with destination resource access capabilities. It is discovered, however that the IAT can be modified so that any call to an API can be redirected to a function within the MPC. The technique for modifying the IAT is documented within the MSDN (Microsoft Developers Network) Library CD in several articles. The technique is also different for each operating system (e.g. between Windows 9× and Windows NT), which can be accommodated by agent generator configurability, such as that given above. MPC **341** therefore has at least initial access to API IAT **731** of Downloadable **632**, and provides for diverting, evaluating and responding to attempts by Downloadable **632** to utilize system APIs **731**, or further in accordance with protection policies **342**. In addition to API diverting, MPC **341** can also install filter drivers, which can be used for controlling access to resources such as a Downloadable-destination file system or registry. Filter driver installation can be conducted as documented in the MSDN or using other suitable methods.

Turning to FIG. **8** with reference to FIG. **7***b*, an MPC **341** according to an embodiment of the invention includes a package extractor **801**, executable installer **802**, sandbox engine installer **803**, resource access diverter **804**, resource access (attempt) analyzer **805**, policy enforcer **806** and MPC de-installer **807**. Package extractor **801** is initiated upon initiation of MPC **341**, and extracts MPC **341** elements and protection policies **342**. Executable installer **802** further initiates installation of a Downloadable by extracting the downloadable from the protected package, and loading the process into memory in suspended mode (so it only loads into memory, but does not start to run). Such installation further causes the operating system to initialize the Downloadable's IAT **731** in the memory space of the downloadable process, P2, as already noted.

Sandbox engine installer **803** (running in process space P1) then installs the sandbox engine (**803**–**805**) and policies **342** into the downloadable process space P2. This is done in different way in each operating system (e.g. see above). Resource access diverter **804** further modifies those Downloadable-API IAT entries that correspond with protection policies **342**, thereby causing corresponding Downloadable accesses via Downloadable-API IAT **731** to be diverted resource access analyzer **805**.

During Downloadable operation, resource access analyzer or "RAA" **805** receives and determines a response to diverted Downloadable (i.e. "malicious") operations in accordance with corresponding protection policies of policies **342**. (RAA **805** or further elements, which are not shown, can further similarly provide for other security mechanisms that might also be implemented.) Malicious operations can for example include, in a Windows environment: file operations (e.g. reading, writing, deleting or renaming a file), network operations (e.g. listen on or connect to a socket, send/receive data or view intranet), OS registry or similar operations (read/write a registry item), OS operations (exit OS/client, kill or change the priority of a process/thread, dynamically load a class library), resource usage thresholds (e.g. memory, CPU, graphics), etc.

Policy enforcer **806** receives RAA **805** results and causes a corresponding response to be implemented, again according to the corresponding policies. Policy enforcer **806** can, for example, interact with a user (e.g. provide an alert, receive instructions, etc.), create a log file, respond, cause a response to be transferred to the Downloadable using "dummy" or limited data, communicate with a server or other networked device (e.g. corresponding to a local or remote administrator), respond more specifically with a better known Downloadable, verify accessibility or user/

system information (e.g. via local or remote information), even enable the attempted Downloadable access, among a wide variety of responses that will become apparent in view of the teachings herein.

The FIG. 9 flowchart illustrates a protection method according to an embodiment of the invention. In step **901**, a protection engine monitors the receipt, by a server or other re-communicator of information, and receives such information intended for a protected information-destination (i.e. a potential-Downloadable) in step **903**. Steps **905–911** depict an adjunct trustworthiness protection that can also be provided, wherein the protection engine determines whether the source of the received information is known to be "unfriendly" and, if so, prevents current (at least unaltered) delivery of the potential-Downloadable and provides any suitable alerts. (The protection engine might also continue to perform Downloadable detection and nevertheless enable delivery or protected delivery of a non-Downloadable, or avoid detection if the source is found to be "trusted", among other alternatives enabled by the teachings herein.)

If, in step **913**, the potential-Downloadable source is found to be of an unknown or otherwise suitably authenticated/certified source, then the protection engine determines whether the potential-Downloadable includes executable code in step **915**. If the potential-Downloadable does not include executable code, then the protection engine causes the potential-Downloadable to be delivered to the information-destination in its original form in step **917**, and the method ends. If instead the potential-Downloadable is found to include executable code in step **915** (and is thus a "detected-Downloadable"), then the protection engine forms a sandboxed package in step **919** and causes the protection agent to be delivered to the information-Destination in step **921**, and the method ends. As was discussed earlier, a suitable protection agent can include mobile protection code, policies and the detected-Downloadable (or information corresponding thereto).

The FIG. **10***a* flowchart illustrates a method for analyzing a potential-Downloadable, according to an embodiment of the invention. As shown, one or more aspects can provide useful indicators of the inclusion of executable code within the potential-Downloadable. In step **1001**, the protection engine determines whether the potential-Downloadable indicates an executable file type, for example, by comparing one or more included file headers for file type indicators (e.g. extensions or other descriptors). The indicators can be compared against all known file types executable by all protected Downloadable destinations, a subset, in accordance with file types executable or desirably executable by the Downloadable-destination, in conjunction with a particular user, in conjunction with available information or operability at the destination, various combinations, etc.

Where content analysis is conducted, in step **1003** of FIG. **10***a*, the protection engine analyzes the potential-Downloadable and determines in accordance therewith whether the potential-Downloadable does or is likely to include binary information, which typically indicates executable code. The protection engine further analyzes the potential-Downloadable for patterns indicative of included executable code in step **1003**. Finally, in step **1005**, the protection engine determines whether the results of steps **1001** and **1003** indicate that the potential-Downloadable more likely includes executable code (e.g. via weighted comparison of the results with a suitable level indicating the inclusion or exclusion of executable code). The protection engine, given a suitably high confidence indicator of the inclusion of executable code, treats the potential-Downloadable as a detected-Downloadable.

The FIG. **10***b* flowchart illustrates a method for forming a sandboxed package according to an embodiment of the invention. As shown, in step **1011**, a protection engine retrieves protection parameters and forms mobile protection code according to the parameters. The protection engine further, in step **1013**, retrieves protection parameters and forms protection policies according to the parameters. Finally, in step **1015**, the protection engine couples the mobile protection code, protection policies and received-information to form a sandboxed package. For example, where a Downloadable-destination utilizes a standard windows executable, coupling can further be accomplished via concatenating the MPC for delivery of MPC first, policies second, and received information third. (The protection parameters can, for example, include parameters relating to one or more of the Downloadable destination device/process, user, supervisory constraints or other parameters.)

The FIG. **11** flowchart illustrates how a protection method performed by mobile protection code ("MPC") according to an embodiment of the invention includes the MPC installing MPC elements and policies within a destination device in step **1101**. In step **1102**, the MPC loads the Downloadable without actually initiating it (i.e. for executables, it will start a process in suspended mode). The MPC further forms an access monitor or "interceptor" for monitoring or "intercepting" downloadable destination device access attempts within the destination device (according to the protection policies in step **1103**, and initiates a corresponding Downloadable within the destination device in step **1105**.

If, in step **1107**, the MPC determines, from monitored/intercepted information, that the Downloadable is attempting or has attempted a destination device access considered undesirable or otherwise malicious, then the MPC performs steps **1109** and **1111**; otherwise the MPC returns to step **1107**. In step **1109**, the MPC determines protection policies in accordance with the access attempt by the Downloadable, and in step **1111**, the MPC executes the protection policies. (Protection policies can, for example, be retrieved from a temporary, e.g. memory/cache, or more persistent storage.)

As shown in the FIG. **12***a* example, the MPC can provide for intercepting Downloadable access attempts by a Downloadable by installing the Downloadable (but not executing it) in step **1201**. Such installation will cause a Downloadable executor, such as a the Windows operating system, to provide all required interfaces and parameters (such as the IAT, process ID, etc.) for use by the Downloadable to access device resources of the host device. The MPC can thus cause Downloadable access attempts to be diverted to the MPC by modifying the Downloadable IAT, replacing device resource location indicators with those of the MPC (step **1203**).

The FIG. **12***b* example further illustrates an example of how the MPC can apply suitable policies in accordance with an access attempt by a Downloadable. As shown, the MPC receives the Downloadable access request via the modified IAT in step **1211**. The MPC further queries stored policies to determine a policy corresponding to the Downloadable access request in step **1213**.

The foregoing description of preferred embodiments of the invention is provided by way of example to enable a person skilled in the art to make and use the invention, and in the context of particular applications and requirements thereof. Various modifications to the embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles, features and teachings disclosed herein. The embodiments described herein are not intended to be exhaustive or limiting. The present invention is limited only by the following claims.

What is claimed is:

1. A processor-based method, comprising:

receiving downloadable-information;

determining whether the downloadable-information includes executable code; and

causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code,

wherein the determining comprises performing one or more analyses of the downloadable-information, the analyses producing detection-indicators indicating whether a correspondence is detected between a downloadable-information characteristic and at least one respective executable code characteristic, and evaluating the detection-indicators to determine whether the downloadable-information includes executable code.

2. The method of claim **1**, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence.

3. The method of claim **1**, wherein the evaluating includes assigning a weighted level of importance to at least one of the indicators.

4. A processor-based method, comprising:

receiving downloadable-information;

determining whether the downloadable-information includes executable code; and

causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code,

wherein the causing mobile protection code to be communicated comprises forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be communicated to the at least one information-destination.

5. The method of claim **4**, wherein the sandboxed package is formed such that the mobile protection code will be executed by the information-destination before the downloadable-information.

6. The method of claim **5**, wherein the sandboxed package further includes protection policies according to which the mobile protection code is operable.

7. The method of claim **6**, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is received before the downloadable-information, and the downloadable information before the protection policies.

8. The method of claim **6**, wherein the protection policies correspond with at least one of the information-destination and a user of the information destination.

9. A processor-based system, comprising:

an information monitor for receiving downloadable-information;

a content inspection engine communicatively coupled to the information monitor for determining whether the downloadable-information includes executable code; and

a packaging engine communicatively coupled to the content inspection engine for causing mobile protection code ("MPC") to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code,

wherein the content inspection engine comprises one or more downloadable-information analyzers for analyzing the downloadable-information, each analyzer producing therefrom a detection indicator indicating whether a downloadable-information characteristic corresponds with an executable code characteristic, and an inspection controller communicatively coupled to the analyzers for determining whether the indicators indicate that the downloadable-information includes executable code.

10. The system of claim **9**, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence.

11. The system of claim **9**, wherein the evaluating includes assigning a weighted level of importance to at least one of the detection-indicators.

12. A processor-based system, comprising:

an information monitor for receiving downloadable-information;

a content inspection engine communicatively coupled to the information monitor for determining whether the downloadable-information includes executable code; and

a packaging engine communicatively coupled to the content inspection engine for causing mobile protection code ("MPC") to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code,

wherein the packaging engine comprises an MPC generator for providing the MPC, a linking engine coupled to the MPC generator for forming a sandbox package including the MPC and the downloadable-information, and a transfer engine for causing the sandbox package to be communicated to the at least one information-destination.

13. The system of claim **12**, wherein the packaging engine further comprises a policy generator communicatively coupled to the linking engine for providing protection policies according to which the MPC is operable.

14. The system of claim **13**, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is executed before the downloadable-information.

15. The system of claim **14**, wherein the protection policies correspond with policies of at least one of the information-destination and a user of the information destination.

16. A processor-based method, comprising:

receiving, at an information re-communicator, downloadable-information, including executable code; and

causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code, wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.

17. The method of claim **16**, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted.

18. A sandboxed package formed according to the method of claim **17**.

19. The method of claim **17**, wherein the forming comprises generating the mobile protection code, generating the sandboxed package, and linking the mobile protection code, protection policies and downloadable-information.

20. The method of claim **19**, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination.

21. The method of claim **20**, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.

22. A sandboxed package formed according to the method of claim **16**.

23. The method of claim **16**, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator.

24. The method of claim **16**, wherein the re-communicator is at least one of a firewall and a network server.

25. The method of claim **16**, wherein the sandboxed package has a same file type as the downloadable-information, thereby causing the mobile code executor to be unaware that the protected package is not a normal downloadable.

26. The method of claim **25**, wherein the sandboxed package is formed using concatenation of a mobile protection code, a policy, and a downloadable.

27. The method of claim **16**, wherein executing the mobile protection code at the destination causes downloadable interfaces to resources at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

28. A processor-based system, comprising:

receiving means for receiving, at an information re-communicator, downloadable-information, including executable code; and

mobile code means communicatively coupled to the receiving means for causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code,

wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.

29. The system of claim **28**, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted.

30. The system of claim **29**, wherein the forming comprises generating the mobile protection code, generating the protection policies, and linking the mobile protection code, protection policies and downloadable-information.

31. The system of claim **30**, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination.

32. The system of claim **31**, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.

33. The system of claim **28**, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator.

34. The system of claim **33**, wherein the re-communicator is at least one of a firewall and a network server.

35. The system of claim **34**, wherein executing the mobile protection code at the destination causes downloadable interfaces a resource at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

* * * * *